Cranfield University

College of Aeronautics
Department of Aerospace Technology

PhD Thesis

Academic years: 1995-1996

Hippokrates A. Hadjiilias

The Aerodynamic Design and Optimization of a Wing-Fuselage Junction Fillet as Part of a Multi-Disciplinary Optimization Process during the early Aircraft Design Stages

Supervisor: Prof. John P. Fielding

June 1996

This thesis is submitted in partial fulfilment for the degree of Doctor of Science

Abstract

The Aerodynamic Design and Optimization of a Wing–Fuselage Junction Fillet as Part of a Multi–Disciplinary Optimization Process during the Early Aircraft Design Stages

by

Hippokrates A. Hadjiilias

Cranfield University

Professor John P. Fielding, Supervisor

An attempt to minimize interference drag in a wing-fuselage junction by means of inserting a fillet is presented in this thesis. The case of a low-wing commercial transport aicraft at cruise conditions is examined. Due to the highly three dimensional behaviour of the flow field around the junction, a thin-layer Navier-Stokes code was implemented to estimate the drag forces at the junction. Carefully selected design variable combinations based on the theory of Design of Experiments constituted the initial group of feasible cases for which the flow solver had to be run. The drag values of these feasible cases were then used to create a second order response surface which could predict with reasonable accuracy the interference drag given the value of the design variables within the feasible region. A further optimization isolated the minimum interference drag combination of design variable values within the design space. The minimum interference drag combination of design variable values was evaluated numerically by the flow solver. The prediction of the response surface and the numerical value obtained by the flow solver for the interference drag of the optimal wing–fuselage combination differed by less than five percent.

To demonstrate the ability of the method to be used in an interdisciplinary analysis and optimization program, a landing gear design module is included which provides volume constraints on the fillet geometry during the fillet surface definition phase.

The Navier Stokes flow analyses were performed on the Cranfield Cray supercomputer. Each analysis required between eight to twelve CPU hours, and the total CPU time required for the optimization of the six variable model described in the thesis required thirty Navier Stokes runs implementing the Design of Experimens and Surface Response Methodology implementation. For comparison, a typical optimization implementing a classical conjugate directions optimizer with no derivative information available would probably require more than forty iterations.

Both the optimization and the flow solver results are discussed and some recommendations for improving the efficiency of the code and for further applications of the method are given.

To my parents, Alexandros and Konstantinia,

for their enormous support throughout

this last decade

Contents

List of Figures

List of Tables

Acknowledgements

I want to thank my supervisor, Prof. J.P. Fielding, without whose support and advice I would not have been able to complete this thesis. I would also like to thank Prof. Mark Drela for his constructive comments at the beginning of my research. Dr. N. Qin kindly provided information and guidance during my familiarization stage with the Navier-Stokes solver. I would like to aknowledge Mr. Howard Smith's help in obtaining the EAGLE package and providing me with the indispensible program manuals.

I would also like to acknowledge the contribution of my colleague Mr. V. Valsamakis for his help in carrying out some of the most tedious tasks of the simulation analyses.

Additionally I would like to thank the Alexander S. Onassis Foundation for their trust in my abilities and their support throughout the course of my studies at Cranfield.

Finally, my hearty thanks to my parents for their support throughout my work.

This thesis was partly supported by a grant from the Alexander S. Onassis Foundation (Grant Ref. Number O-12-2)

Chapter 1

Introduction

1.1 Background

The design of an aircraft encompasses interactions between a number of disciplines of aeronautical engineering, i.e. structures, aerodynamics, controls, and propulsion (Figure 1.1[131]). Some aspects of design are analytical, while others are too complicated to be calculated analytically and therefore methods that are either empirical, numerical or experimental are used. The aircraft design process is itself a practical experience which determines how the various disciplines should affect each other in order to produce an optimum design while conforming to the requirements.

1.2 The Aircraft Design Process

The aircraft design process can be divided into three phases, the conceptual design, the preliminary design and the detailed design[68].

The conceptual design of the aircraft produces an initial configuration (the general layout of the aircraft, consisting of the external shape, the dimensions, and other features and characteristics like wing type and location, tail surfaces

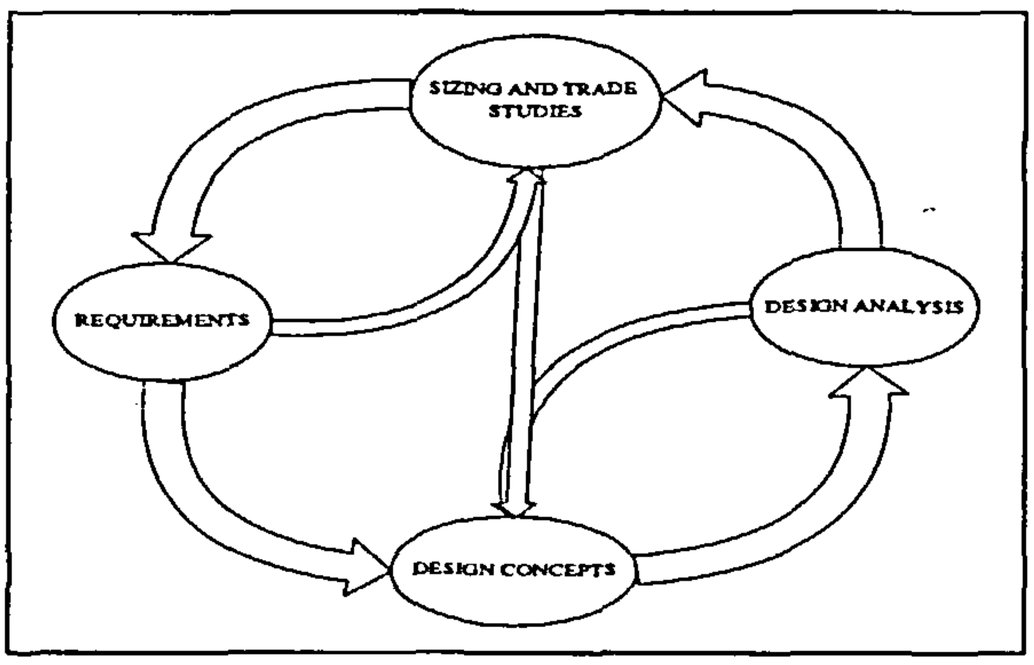

Figure 1.1: Aircraft Design Cycle

type and location), size, a weight estimate and an estimate of the aircraft's performance. At this stage, there is very little information available and the analysis, therefore, only deals with the above general aspects of the design. This is appropriate at this stage, while more detailed analyses will be performed at a later stage. Furthermore, a number of different requirements imposed by the various disciplines must be satisfied; the primary selection of the aircraft's conceptual shape and characteristics are chosen as the best compromise that satisfies the above requirements. By the end of the conceptual design stage, major decisions have been made about the aircraft concerning the fuselage, wing, engines, landing gear, tail surfaces and their positions relative to each other.

The preliminary design usually accepts the results of the conceptual design as fixed, allowing though for minor modifications usually in areas to which the overall design is not particularly sensitive. During this design stage, the components are evaluated in more detail, and components that have not been assessed in adequate detail during the conceptual design stage are now specified and evaluated.

As the process enters the detailed design stage, the analysis performed becomes more involved partly because all the major components and features

have been fixed, and partly because secondary components are beginning to be taken into consideration. The detailed design requires extensive commitment, and therefore is usually performed by different departments according to discipline. During this stage, secondary components are designed and analyzed and system testing becomes increasingly important.

1.3 Interference Drag in Preliminary Design

In the traditional preliminary design stages, although major decisions are being taken concerning the wing and the fuselage, their shape, type and characteristics, and extensive simulations are being performed either in wind-tunnels or numerically, aiming to reducing the overall drag, limited design effort is being invested in accounting for "interference[1]" drag, even though in cases, it may account for up to an additional twenty percent on the drag estimates of the separated wing and fuselage each measured in free flow. Some experimental attempts implementing fillets at the wing–root junction indicate the benefits of using a fillet, and suggest a number of configurations (such as Jupp's description of the A310's wing–root fillet [86], Bernstein's strake–like fillet [12]). However, there is rarely any systematic rather than "let's just try it" fillet design methodology suggested in the literature.

With careful design, the interference drag may decrease typically about ten to thirty percent with the insertion of a junction fillet [77].

The main reason for not including the interference drag analysis in the preliminary design stages was the cost and effort it involves. Wind–tunnel testing was too time–consuming and expensive, since a large number of configurations had to be constructed, and there had been no methodology available to de-

[1]Hoerner [77] defines interference drag as the difference between the combined drag of two merged bodies and the sum of their individual drag components. This interference drag in most practical situations is positive.

rive the shape of any junction fillets. Performing a numerical analysis was also exceedingly expensive, requiring a Navier–Stokes code and necessarily a supercomputer. But even then, there are two main problems with the implementation of a numerical simulation to predict the interference drag and to optimize the shape of the junction fillet at the wing–fuselage junction:

- The time required to evaluate the interference drag for one configuration
- The number of configurations required to produce meaningful conclusions and thus produce an optimum or near-optimum design

This thesis addresses these problems and proposes a methodology for interference drag trend determination at the junction of the wing and the fuselage, based on a combined implementation of a Navier–Stokes computer code, and an optimization method consisting of both statistical (Design of Experiments and Response Surface Methodology) and analytical techniques.

1.4 Introduction to CFD

The development of high–speed computers has had significant impact on the application of the fluid dynamics principles to problems of design in modern aerospace engineering practice. Problems that twenty years ago required years to work out, can be solved today within a few minutes. This availability of such computer power has caused many changes, first noticeable in industry and research laboratories, where the need to solve complex problems was the most urgent.

Thus, a new methodology for attacking the complex problems in fluid mechanics became increasingly more important and was termed Computational Fluid Dynamics (CFD). In this computational approach, the equations, usually in partial differential form, governing a phenomenon of interest are solved

numerically. Some of the solution methods were known at the turn of the century. Richardson [132], in 1910, calculated stress distributions on dam applications by using point iterative methods to solve the Laplace equation. And in 1928, Courant, Friedrichs, and Lewy [38] addressed the numerical solutions of partial differential equations. But the evolution in computational activity did not appear until the general availability of high-speed digital computers, an event which occurred in the late 1950's.

Traditionally, both experimental and theoretical methods were the two principal ways implemented to develop designs for equipment and vehicles involving fluid flow governed by non–linear equations. With the development of the digital computer, the numerical approach has become available. A great number of significant contributions from researchers has propelled the development of methodologies beyond any early expectations. In this introduction it is not possible to mention all the CFD contributions that have taken place since the 1920's. An attempt, therefore, is made to mention only the most significant contributions to CFD since its beginnings.

A major contribution to CFD is due to John von Neumann who developed the stability criteria for numerical methods used for solving time–marching problems [122] which was the first practical tool available to researchers to ascertain numerical methods.. Another significant contribution came in 1954 from Lax [96] who developed a shock capturing method for computing flows with shocks by using the conservation-law form of the governing equations. At nearly the same time Frankel [58] presented the successive overrelaxation method for the first time, applied to Laplace's equation. In 1957 Richtmyer [133] and later in 1967 Richtmyer and Morton [134] provided a compilation of tools for the solution of marching problems. At the same time Douglas and Rachford [46] developed a number of implicit methods applicable to parabolic and elliptic equations. Forsythe and Wasow [56] provided a compilation of

methods for elliptic problems.

In the 1960's a number of important developments occured, involving the solution of flows with discontinuities. Artificial viscosity schemes were developed [154], and in 1960 Lax and Wendroff [97] published their method for computing flows with discontinuities. MacCormack [102] in 1969 developed his version of the Lax and Wendroff method, providing one of the most popular techniques for solving problems involving flow discontinuities.

In 1967 Richtmyer and Morton [134] produced a theoretical structure for analyzing copmutational methods for fluid dynamics and review a number of finite difference methods for inviscid compressible flow. Finite difference techniques are also used by Roache [136] to analyse viscous separated incompressible and compressible flow.

Although experimentation continues to be important, especially when the flows involved are very complex, the trend is clearly toward greater reliance of computer based predictions in design. This point however has promoted a debate between the experimentals and the computer modellers. The first such confrontation appeared in the 1975 paper by Chapman, Mark and Pirtle [32]. The paper identified the main limitations of computer modelling at the time, i.e. the computing speed, and the lack of efficient methods. Since then great improvments have taken place both in terms of computational speed and efficiency of methods. In Chapman's 1979 paper [30] is shown that over a ten year period, two orders of magnitude improvement was achieved in both the areas of computing speed and efficiency of methods used [30]. Chapman [31], Green [67], Krause [92] and Jameson [83] provide a sample of what engineering CFD was capable of at the time, and what could be expected of it in the future.

More recently Peyret and Taylor [125] provided an overview of CFD techniques with emphasis on finite difference and spectral methods. Efficient techniques for boundary layer flow and inviscid compressible flow are presented

by Holt [79]. More recently Fletcher [55] provides techiques for finite element and spectral methods. Current state of the art methodologies take advantage of vector and parallel computers. Ortega and Voigt [124] and Gentzsch and Neves [64] offer a comprehensive introduction of this area.

Three methods for solving the governing equations for a flow problem are available to the researcher. They are: panel methods, finite differences and finite volumes.

For cases that can be approximated by linear differential equations (e.g. the Prandtl-Glauert or Laplace's equation) a panel method may produce acceptable results. Panel methods work on the principle of superposition: Since the equations solved are linear, it is possible to multiply a known solution by a scalar and add these reults to form more general solutions. This can be made to work in both subsonic and supersonic cases. Sources or doublets or vortices of either constant or varying strength are located in the flow so that their combined solutions satisfy the boundary conditions of the problem. The boundary conditions are typically that the combined flow does not go through the surface, and that far from the body, the flow approaches the freestream solution.

Nonlinear CFD methods are used to predict complex flow fields such as transonic or separated flows. The basic equations can be expressed either as finite difference or finite volume. For the finite difference form, the equation is discritized in time and space, while for the finite volume the integral formulation of the partial differential equation is discretized. In terms of the grids used, the finite difference method will calculate solutions at all the grid node points, while the finite volume method will compute the solution at the cell centroids. There are a number of grid types available, and are mainly categorized under either structured grids or unstructured grids.

The typical approach, for non-linear problems is to either implement a

complex geometry and a simplified flow model, or use a complex flow model with a simplified geometry, both consisting of either a finite volume or a finite difference discretization of the differential form of the governing equations. This is due once more to computational expense and extravagant storage and computational time requirements. Lately, this approach is becoming less compromising and explores methods and concepts that will allow exploration of complex geometries concurrently with the use of complex flow models. When this is achieved, the next step is to use these methods for vehicle or component design and optimization, even from the preliminary stage, blurring, hopefully irrevocably, the lines between conceptual, preliminary and detailed design.

On the way of the researcher towards this goal, one of the most challenging problems that arises is the uncompromising coexistence of the large computational time demands of a complex flow – complex geometry numerical solution, and the prohibitive number of such runs dictated by a multi–variable optimization. This thesis presents an attempt to implement a thin-layer Navier-Stokes solver within an optimization loop used for the design of a wing–fuselage fillet as it is constrained by a number of different aircraft systems requirements. The paramount consideration for such a task is to reduce the computational time as far as possible, and this reduction, in turn, promotes the reduction of the number of the computational runs. The need for such a strict control over the number of computational runs during the optimization process was addressed by the implementation of a method that has been known since the 1940's and has been refined ever since: the method of Experimental Design.

1.5 Introduction to the Design Of Experiments

The method of *Design of Experiments* primarily deals with situations where the goal of a simulation study is not defined clearly: one may wish to find

out which of possibly many parameters and structural assumptions have the greatest effect on a performance measure, or which set of model specifications appears to lead to optimal performance[95].

In simulation, *experimental design* provides a way of deciding before the simulation or experimental runs are made, which particular configuration to simulate, so that the desired information can be obtained with the least amount of simulating. Carefully designed experiments or runs are much more efficient than a "hit-or-miss" sequence of runs in which a number of alternative configurations are input usually chosen in random.

As the behaviour of the model becomes more obvious (in particular which factors really matter and how they appear to be affecting the model's response) it is typical to move on and become more precise; for instance one often seeks optimal combinations of features or characteristics that minimize or maximize the response of the model (e.g. minimizing the interference drag of the wing–fuselage junction). In this case, an entire variety of statistical techniques known as *metamodelling* and *Response Surface Methodologies* can be used to attain such goals. Such techniques are usually complemented by the related topic of *gradient estimation and sensitivity*, where the way the model responses react to small changes in the quantitative factors, features or characteristics is quantified.

Therefore, what the combined implementation of the experimental design methods along with the response surface methods offer is an initial exploration of the design space, an adequate approximation of this space within a predetermined number of experimental runs, and the formulation of an analytical approximation model of the design space in the guise of a response surface. It is a very powerful tool but is also subject to a number of assumptions that must be kept in mind throughout the analysis. Obviously, not all design spaces can be adequately described by a certain number of experimental evaluations.

Also, not all approximations will be appropriate to the design space; the model used for the approximation will determine the number of experiments required and also the "goodness" of the approximation fit to the design space. This influences directly the efficiency of the response surface as a stepping stone towards the location of the optimum region, and therefore the overall efficiency of the optimization scheme to be used.

1.6 A View on Optimization

In the previous two sections, the two main tools for the computational analysis and approximation of the design space properties were introduced. They involve a very intense computational effort, which results in the definition of a response surface which contains the information on the optimum region of the design space with respect to minimizing the drag of the wing-fuselage junction fillet. Throughout the analysis, though, it is of great importance not to lose the perspective of the entire effort, which is to intelligently interpret the results rather than to simply obtain results, and to utilize the methods as an aid to discover new kinds of design solutions.

This implies that the maturation of computerized design methods will not likely revolutionize design or designers' practices. It will change some of the ways in which they do things, but the advances will always come from the conceptual and the creative rather than from the repetitive and the mechanical. In this spirit, the aims of the entire optimization effort is to identify not only an optimal region or location, but to mainly describe and characterize the design space that consists of all the feasible configurations within the constraints imposed. Richard Hamming [70] says in the motto to his book on numerical methods: "The purpose of computing is insight, not numbers."

1.7 Motivation

One of the most challenging problems in computational fluid dynamics is the accurate prediction of the junction flow properties at the wing–fuselage junction, which in design terms correspond to the prediction of the interference drag of the junction. To date, the wing and tail junction fillets are designed by experience and wind–tunnel cut–and–try. Even the different types of Navier–Stokes codes must be chosen with care just to ensure that they can predict drag reliably enough for this type of flow to permit any kind of significant numerical optimization.

The prediction of the interference drag at the wing–fuselage junction becomes very important when one considers that in a case of massive corner separation the additional drag of the corner is up to 30% of the sum of the drag of each of the fuselage and wing when measured alone in the freestream. The actual drag estimate can be obtained either by using a local or a field method [3] [115]. The first derives a drag estimate by integrating the scin friction and the notmal pressure in the streamwise direction, while the second method evaluates the drag by evaluating the "far field integral" which consists of a wave, a viscous and a vortex component.

But even the term "interference drag" is unfortunate since it often indicates that the real mechanism is not understood. After all, drag is either profile form drag, skin friction drag, induced drag or wave drag. The additional drag due to the corner is most likely form drag, although induced drag could also be important if the spanwise lift distribution is adversely altered.

Today, empirical fillet shapes are utilized in the junction region, mainly for mass considerations (such as lack of sufficient volume for various systems like electronics, undercarriage, etc.) Wing root fillets can also significantly affect the junction flow at the wing-fuselage jucntion area, as demonstrated by

Jupp [86]. However, Jupp admits that using theoretical methods to predict a favorable fillet design was too costly in terms of effort and time.

It is evident that a method capable of optimizing the shape of a fillet with the goal of reducing the interference drag will produce a positive benefit to the preliminary design and optimization of the entire aircraft, by taking into account several aircraft system interactions and accounting for any extra space that may be available by the introduction of such a fillet.

The incorporation of such an elaborate method of drag and shape prediction and optimization respectively into the preliminary design stages should result in the blurring of the distinction between preliminary and detailed design, which, in its turn, should drastically reduce the transition time between the two design stages essentially by integrating the two. In addition, it should address the need for an efficient, easy to use, reliable and modular method to evaluate preliminary design aircraft configurations and thus further reduce the time required for the preliminary and detailed design and evaluation of an aircraft.

1.8 Thesis Outline

The first chapter of this dissertation presents the background that the rest of the thesis builds upon. The information given is very general, and it places the contents of the thesis in perspective to historical as well as current developments. The goal of the dissertation is mentioned, and an introduction to the basic tools implemented to achieve it is presented. The reasons for undertaking the research are also outlined.

The second chapter presents the preliminary investigation that was accomplished before the main research began. The material in this chapter addresses the inadequacies of the presently popular methods of analysis and optimization

to address the problem of optimizing the shape of the wing–fuselage junction fillet with respect to the interference drag. This problem can be generalized to any flow field problem requiring optimization, where the cost of a single experimental or computational run is prohibitively high. Also in this chapter the reasons for deciding on the tools that were eventually used become more apparent.

In Chapters three through five the main analytical and numerical analysis tools are described. They consist of an analysis module based on the statistical methods of Design of Experiments and Response Surface Methodology, a CFD modeling package, a thin–layer Navier–Stokes solver, and a Landing Gear Analysis Module.

The statistical methods of Design of Experiments and Response Surface Methodology, while being widely used in non-aerospace industrial applications, are rarely used in terms of preliminary aerospace vehicle design. The fourth chapter attempts to present these two alternate approaches as significant alternatives to classical design and optimization techniques. Within the CFD modeling framework, a new method based on old techniques of surface representation is described and implemented for the geometry definition input to the solver. The properties, advantages and disadvantages of the thin–layer Navier–Stokes code are presented, along with the results and conclusions from its use. The Landing Gear Analysis Module consists of preliminary landing gear design estimates, and is used to ensure that the landing gear will fit under the wing-fuselage shape.

Having defined the tools, the next step is to close the loop by introducing the optimization method. This is the point where the unrealistic implementation of a viscous flow field optimization is transformed to a viable method for the preliminary optimization of the generic wing–fuselage configuration under certain assumptions. The emphasis of the method is on trend pre-

diction rather than on the accuracy of a particular solution. In addition to extracting an approximation to the optimum configuration, the method also approximates and describes the function space thus providing invaluable insight to the designer about the sensitivity of the design. The statistical initial approach is complemented by a classical optimization scheme to produce an accurate approximation of the optimum configuration.

The model, its validation, properties, advantages and disadvantages, its use and the results obtained from it are discussed in chapter seven. Finally the conclusions and the recommendations for further work are presented in chapter eight.

Chapter 2

Literature Survey and Preliminary Investigation

The literature survey and the early work relevant to this thesis mainly covers two categories, the requirements for the wing-fuselage junction flow analysis and the statistical methodology consisting of the Design of Experiments and Response Surface Methodology.

2.1 Wing-Fuselage Junction Flow Investigation

Junction flows is one of the most challenging areas in the fields of aerodynamics, in the incarnation of interference drag, and of Computational Fluid Dynamics. A number of different attempts have been made throughout the years to calculate the flow either empirically, analytically, experimentally or computationally around idealized junction geometries and slightly more complex configurations.

2.1.1 Empirical Methods

In preliminary aircraft design the need for an initial estimate of the aircraft's total drag required the development of methodologies to roughly estimate the interference drag for the wing-fuselage junction and also for the tail junctions. A number of different estimation methods appear in classical design handbooks such as Roskam [138], Raymer [131] and Clancy [35]. Some practical guidelines are often mentioned, that facilitate the preliminary configuration design. McCormick [105] notes that drag penalty at the wing-fuselage junction will become more severe if surfaces meet at an angle other than ninety degrees. In particular, he continues, acute angles between intersecting surfaces should be avoided. Many sources mention that in the case of acute angles, if they cannot be avoided then filleting should be used.

The available data on wing-fuselage interference drag are sparse. Usually, no correlation with wing position or lift coefficient is made [77]. Even then, while such data may be helpful in estimating interference drag, an accurate estimate of this quantity is nearly impossible. The most detailed description of practical methodologies available for the preliminary prediction of the interference drag along with preliminary configuration guidelines are presented by Hoerner [77]. On the interference drag arising from an idealized wing-flat plate junction, Hoerner notices that two main dimensions influence the magnitude of the interference drag, both dimensions of the wing section: the "chord area" c^2 and the "thickness area" t^2. He presents an overview on the effect of the boundary-layer thickness and also on the influence of lift on the drag magnitude: *Boundary Layer Thickness* Hoerner [77] expects the interference drag of the wing-wall junction to be a function of the boundary layer at the wall. Here his mention for negative interference drag arises, for typical thickness ratios below $t/c \approx 0.08$. *Influence of Lift* Hoerner suggests that the lift from the wing may produce an increased pressure gradient on the upper side

of the wing-fuselage function. He then claims that the interference drag increases with the lift coefficient, as in the case of the NACA 0012 airfoil, where Hoerner shows that the interference drag increased "as the square of the lift coefficient".

Hoerner also considers the advantageous use of fillets on wing-fuselage and tail plane junctions. He only considers simple radial fillets, and suggests that the optimal radius is small, in the order of 4% to 8% of the wing chord /citeHoerner. He also stresses that with fillets extending beyond the trailing edges of the wings interference drag can be reduced significantly, while many researchers place the emphasis of fillet design on the leading edge fillet extensions [12] [86] [33].

2.1.2 Theoretical Methods

Because of the complexity of the wing-fuselage junction problem, and the strong non-linear effects occuring at transonic speeds, not many direct theoretical methods exist without having to resort to at least some CFD methodologies. An overview of early theoretical formulations can be found in Ferrari's review [54]. Ashley and Landahl [4] present a generalized methodology based on potential flow, implementing singularities. Since it is a linear theory, it is possible to model the initial complex configuration as a system of aerodynamic objects and superimpose the disturbance each one of them causes when placed in the flow. Then the singularity strengths and distributions are varied for each object until the boundary conditions are satisfied (e.g. no cross-flow through surfaces). Ashley and Landahl [4] reasonably point out that this method can only be used as a theoretical tool where either the interactions between no more than two surfaces are not too strong, or where the interactions are "unidirectional" i.e. where only the first of the two surfaces influences the second.

Obviously using the method of superimposing linear solutions and modi-

fying the singularity attributes to estimate the flow around the wing-fuselage junction will require an iterative process[4]. It is much preferable at this point to use a CFD methodology (panel methods in this case of linear theories) rather than attempt to carry out the iterative process theoretically. The more complex the model is, the stronger the tendency to use CFD methods to solve a particular configuration after the original theoretical investigation has yielded the appropriate linear or non-linear formulation of the problem. Therefore the researcher can produce results much faster and with much less effort for one specific configuration at predetermined conditions, than for a generic condition regime and a configuration family.

Many publications preferred to do just this: specific problems of interference in predetermined geometries. For example, McDevitt and Haire [106] investigated a body-contouring method for alleviating the adverse interference at the root of a sweptback wing at high subsonic speeds. The analytical background was provided by Kuchemann [93] who treats in detail the design of wing-fuselage junctions for subsonic speeds. It was proposed that the body be represented by a cylinder in which ring vortices are distributed so that the induced axial component of velocity cancels the swept-wing interference velocity. By integration of the induced drag velocity, the radial modification necessary to shape the wing-body junction is determined. The latter is a method that is widely used, with some occasional modifications. Experimental results were obtained at R.A.E.[145], while Weber and Joyce [155] [156] developed a modified method to calculate the interference problems on wing-fuselage combinations. The series of reports published summarizes studies of the interference of swept and unswept wings, lifting or at zero incidence attached at a mid-wing position to a cylindrical fuselage. In this case, the sweep of the wing is accounted for by a swept source or vortex line extending inside the fuselage.

Summarizing, the analytical methods usually assume potential flow since they use linear theory which allows superposition. Due to the effort involved, they can only practically handle simple configurations, with the most complex being a swept, lifting mid-wing extending to infinity on a cylindrical fuselage. It is possible to modify the analysis for more complex configurations and even introduce fillets at the junction, but the results would not be acceptable since the regime of interest is transonic flow, and an important factor giving rise to interference drag is the boundary layer interaction at the junction and the analytical methods are unable to handle such considerations since it is believed that viscous phenomena are important.

2.1.3 Computational Methods

Juncture flows involve the study of some very important applications in addition to the wing-body junction, examples of which are the multi-body shuttle configuration and the turbine rotor-stator interactions. In addition to external aerodynamics applications (e.g. wing-body junctions, wing-pylon junction) juncture flows are also important in meteorology and even geology. The common underlying model in all of these applications is that of a generic juncture flow consisting of a cylindrical protuberance mounted normal to a flat plate as shown in Figure 2.1.

Numerous generic juncture flow cases have been studied experimentally ([99], [141], [6], [7], [51], [51], [147]). In these it is evident that the complex horseshoe vortex system and wake structure remain a challenge for researchers. A number of computational papers have attempted to study laminar or turbulent juncture flow for various configurations, such as the work ofPotsdam and Intemann [129], van Muijden et al. [152], Wichmann [158], Rill [135], Kaul, Kwak and Wagner [87], Visbal [153], and Chen and Hung [34]. In Visbal and Chen,the approach is along the following lines: The low speed laminar

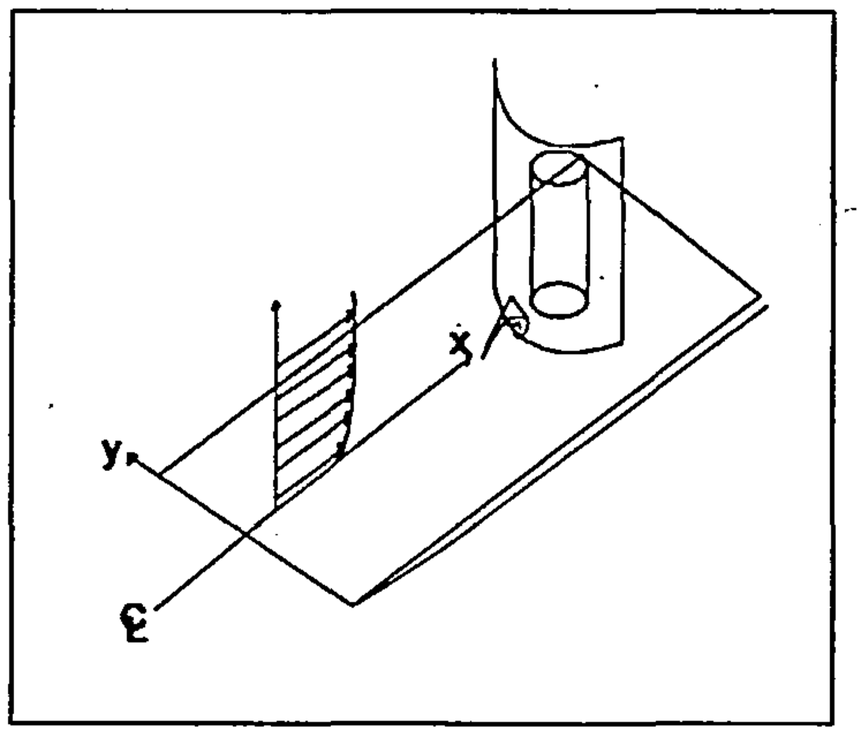

Figure 2.1: Generic Juncture Flow

flows are computed first to determine the Reynolds number effect with the same incoming boundary layer thickness. Then a number of qualitative comparisons are made among the computations, experimental observations, and maybe analytical work.

Finally, Sung, Griffin and Taylor [147] have demonstrated the use of a central difference, finite-volume, explicit Runge-Kutta time-stepping scheme for the solution of steady state, incompressible 3D Reynolds-averaged Navier-Stokes equations to solve the incompressible turbulent horseshoe vortex juncture flow.

The applicability of the Navier-Stokes equations and particularly the efficiency of the simplified versions of these equations is analyzed in a number of Computational Fluid Dynamics textbooks (such as Henne [72], Hirsch [76], Anderson, Tannehill and Pletcher [2]).

2.2 Design of Experiments and Response Surface Methodology

The Design of Experiments is a method that requires a small number of simulations to describe the design space. This is partly due to the fact that the design space is assumed to exhibit a particular behaviour in the region of interest (such as linear or quadratic) and partly to the fact that the choice of design point evaluations is very carefully made. Where the cost of a single design point evaluation is high, minimum point designs are implemented. These designs are constructed so that they will produce a model of the objective to be approximated with the minimum number of design point evaluations appropriate to the assumptions of linear or quadratic behavior. The design point evaluations are then used to produce a response surface, which is an analytical description of the objective function approximation. If the objective was assumed to exhibit linear behavior, then the response surface would be a first order polynomial, while if the assumption was for quadratic behavior, the response surface would be a second order polynomial.

Analytically, experimental strategy and analysis in Response Surface Methodology (RSM) revolves around the assumption that a response η is a function of a set of design variables $x_1, x_2, \ldots, x_k$ where k is the number of variables, and that the function can be approximated in some region of the x's by a polynomial model. Prominent among the models considered are the first-order model:

$$\eta = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k \tag{2.1}$$

and the second order model:

$$\eta = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} x_i^2 + \sum_{i=1}^{k} \sum_{j=1}^{k} \beta_{ij} x_i x_j \qquad i < j, k = no\ of\ variables \tag{2.2}$$

So, an example of a first order response surface approximating some objective

function of two variables x_1 and x_2 would be:

$$\eta(x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \tag{2.3}$$

where $\beta_0, \beta_1, \beta 2$ are constants. Similarly, a second order response surface approximating some objective function of two variables x_1 and x_2 would be:

$$\eta(x_1, x_2) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2 + \beta_{12} x_1 x_2 \tag{2.4}$$

Generally, the assumption of common error variance σ^2 is made, suggesting the use of ordinary least squares for estimation of coefficients.

2.2.1 Response Surface Design

A multitude of publications exists on the Response Surface Design. Hill and Hunter [75] emphasized practical applications in the chemical and processing fields, featuring an excellent bibliography up to 1966. Hill and Hunter [74] stated that the RSM began with the work of Box and Wilson [20]. Mead and Pike [107] moved the origin of RSM back into the 1930s to include the use of "response curves".

Among the many important works before the article by Hill and Hunter, one noteworthy article is the one by Box and Wilson [20] in which the notion of composite designs was introduced. The "star" portion of the design for augmentation purposes of a two-level factorial array was done to allow for efficient estimation of quadratic terms in the second order model of Equation 2.2. Box and Hunter [18] placed emphasis on judging a design on the basis of a prediction variance, var $\hat{y}/\sigma^2$ where $\hat{y}$ is the estimated response and σ^2 is the common error variance. For the first time, the concept of rotatability was introduced; rotatability is achieved when the variance of a predicted value remain constant at points that are equidistant from the design center. In the first-order models, rotatability is achieved with standard orthogonal arrays. In

the case of a second-order model, composite designs and other designs can be made to be rotatable very easily. A third influential publication was by Box and Draper ([21] [22]), which introduced the notion of robustness of an RSM design to model misspecification [1]. Arguments are made that not only can bias due to model misspecification not be ignored but that, if there is even a modest amount of misspecification, the user must seriously consider it in choice of design.

After the fundamentals had been laid out, a number of families of useful experimental designs for first and second order models appeared. In the first-order case, the need for orthogonal designs was motivated by Box and Wilson [20],and Box [15]. Specific design classes, two-level factorial and fractional factorial designs, were made available by Placket and Burman [128]. For second-order models, many scientists and engineers have a working knowledge of the family of central composite designs (CCD) and a class of special three level designs by Box and Behnken [16].

2.2.2 The Central Composite Design

The CCD (see Figure 2.2) belongs to the class of complete designs, where the construction consists of three portions:

1. The 2^k vertices of a cube (or a fraction of these vertices)

2. The $2k$ vertices of a "star" (also known as a *cross-polytope* or *axial portion* with parameter α

[1]The average weighted mean squared error as defined by Box and Draper [22] is: $J = \frac{NK}{\sigma^2}\int_R w(\mathbf{x})E[\hat{y}(\mathbf{x}) - g(\mathbf{x})]^2 d\mathbf{x}$ where $\hat{y}(\mathbf{x})$ is the fitted polynomial of order d_1 and $g(\mathbf{x})$ is a model of order $d_2 > d_1$ containing unknown parameters and is regarded as the true mean response, i.e. the response one chooses to protect agains. R is the region of interest in the design variables. J consists of the sum of the weighted variance and the squared bias. A robust design is one that comes close to minimizing the bias portion of J. A result of this definition is the formalization of the notion of the proper placing of design points in from the boundary of the region of interest when misspecification is possible.

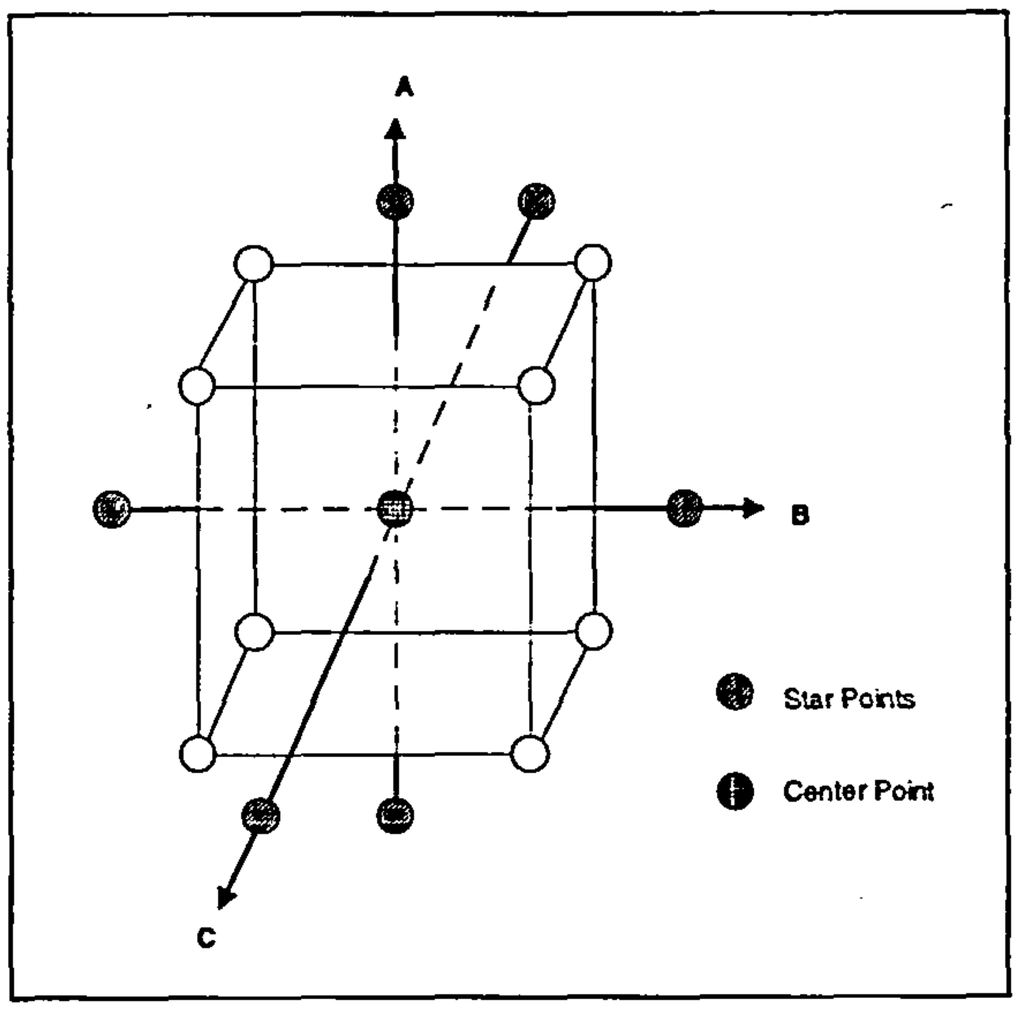

Figure 2.2: Central Composite Design for Three Variables

3. A chosen number of center runs (n_0)

where k is the number of variables, and α is the distance of the axial points from the center of the cube.

The points in (2) essentially form an augmentation that represents a one-factor-at-a-time portion designed to estimate the β_{ii}'s, the diagonal (or pure) quadratic terms in Equation 2.2. The factorial design levels are centered and scaled to design units with ± 1 being the factorial levels and α being the axial level.

The advantages of the CCD is its flexibility and utility as an efficient design plan for sequential experimentation. The factorial portion and the center runs serve as a preliminary phase from which one may fit a first-order model and yet gain evidence regarding the importance of the pure quadratic contributions. The indispensable reading on CCD is provided by the textbooks in RSM by Box and Draper [17], Khuri and Cornell [89], and Myers [117]. The flexibility of the method is provided by allowing the implementor to determine the value

for α and n_0 depending on the application. There are many criteria for the choice of these parameters.

Typically, rotatability is guaranteed by picking an $\alpha = (F)^{1/4}$, where F is the number of factorial points. The experimental region often dictates the value of α. For example, for a spherical region the upper bound for α is usually $\sqrt{k}$. Other criteria for determining α exist, such as robustness to model misspecification, quality estimation of the slope, robustness to outliers, generalized variance of model coefficients, and orthogonal blocking.

The choice of the number of centerpoints is a vital element of the CCD method. Box and Hunter [18] suggests choosing an n_0 for which *uniform information* or *uniform precision* is achieved. This method provides that the value of var $\hat{y}(x)$ is uniform within a sphere of unit radius. Lately, there are indications that fewer center points than initially recommended are generally appropriate. Draper [49] and Box and Draper [17] point out that the initial estimates on the number of center points seem to be somewhat on the large size in light of practical experience.

The CCD is the second-order family that contains candidate designs that *block orthogonally*[2]. Box and Hunter [18] developed the general conditions that give rise to orthogonal blocking in some situations. Examples of orthogonal blocking are given in the books by Khuri and Cornell [89], Box and Draper [17], and Myers [117].

2.2.3 Other Second-Order Designs

Other second-order designs may be efficient in various situations. A low point number second-order designs based on irregular fractions of the 3^n factorial design was proposed by Hoke [78]. These fractions are based on sets of

[2]Orthogonal blocking occurs when regression coefficients are orthogonal to block effects and thus the analysis can be conducted without ambiguous interpretation [18]

partially balanced arrays and are compared to other similar competing designs. Roquemore [137] developed a family of second-order designs called *hybrid designs* for $k = 3, 4, 6$, and 7 that are either saturated or near-saturated[3]and have some similarities to the CCD. The design for k variables can be designed by augmenting a $(k-1)$-dimensional CCD with an additional column in the design matrix. The values for the additional column are chosen to achieve certain design symmetries.

The Hoke hybrid designs are only one of the possible choices that are available when one seeks a saturated or a near-saturated second-order design. Notz [120] suggested a method of constructing designs that are very efficient in terms of generalized variance (also known as D-Efficiency). His designs are saturated and are constructed from the 3^k lattice. Box and Draper [23] produced practical designs that are saturated and efficient from a generalized variance standpoint. Similar developments were given by Mitchell and Bayne [110] and others. During the 1970s and 1980s, much of the work resulting in the development of new second-order designs developed around the concept of D-optimality and D-efficiency. These developments produced usable designs from the design optimality standpoint. It is important for the implementor to understand the tradeoffs when using a saturated or a near-saturated design, i.e. poor coverage of the region of interest.

One important criticism of the optimal-design theory is that is set within a rigid framework governed by a set of assumptions that may not be very realistic. On this, Box and Hunter [18] state that "in recent years, the study of optimal design has become separated from real experimentation with the predictable consequence that its limitations have not been stressed, or, often, even realized".

[3]A design is saturated when it exhibits the same number of design points as terms in the RSM model used (usually linear or quadratic.

In many applications of RSM, good estimation of the derivatives of the response function may be as important as the estimation of the mean response. The computation, of course, of a stationary point in a second-order analysis, or the use of gradient techniques–for example steepest ascent or ridge analysis–depends heavily on the partial derivatives of the estimated response function with respect to the design variables. Since designs that attain certain properties in $\hat{y}$ (estimated response) do not enjoy the same properties for the slopes, it is important for the experimenter to consider designs that are constructed with the derivatives in mind.

Atkinson [5] considered designs for estimation on the slope at a fixed point with the response function being of order 1. The criterion used is the expected mean squared error for a directional derivative averaged over all possible directions. Murty and Studden [116] considered polynomial-regression models with the criterion being the variance of an estimated slope at a fixed point and averaged over an interval.

Hader and Park [69] extended the notion of rotatability to cover the slope for the case of second order models. They catalogued designs that result in *slope rotatability*, i.e. the variance of the estimated derivatives is constant for all points equidistant from the design center. Mukerjee and Huda [114] developed designs associated with minimum variance of the estimated slope maximized over all points in the factor space for second-order and third-order polynomial models over a spherical region.

2.2.4 Applications in Physical and Engineering Sciences

Olivi [123] discussed the need for use of RSM in exploring and identifying certain features of systems. He uses as a "typical" application a study involving five design variables with an orthogonal CCD. The application involves factors that influence ballooning time, an important variable in nuclear safety.

Bodden and Edwards [13] used a Box-Behnken design in an RSM study in the investigation of the mechanism involved in the assay of creatinine. Claycomb and Sullivan [36] used a three factor CCD and a ridge analysis of the data to illustrate the methodology for selecting a cutting tool for maximization of profit.

Contour plots of constant response without an analytic method for finding optimum conditions are often the source of conclusions drawn by RSM analysts. The field of nuclear engineering has not been without studies involving RSM, particularly when simulation is involved. Heller, Oelkers, and Farnswor [71] used data from CCD to study the mechanism involved in thermal-hydraulic margin analysis.

In the field of Aeronautics, very little has been done implementing RSM. A very limited number of publications exist, which mainly demonstrate the applicability of DOE and RSM to a variety of fairly simple test problems.

Such publications include the use of fractional factorial experiments by Sidik [142] where a program was used to implement the Bayes procedure for designing two-level fractional factorial experiments. The Bayes procedure maximized the expected utility over all possible distinct choices of parameter-estimator matchings, physical-design variable matchings, and defining parameter groups for an assumed strategy for nature, when the steps of the design and performance of the experiment are represented as a finite discrete game between the experimenter and nature.

A more practical situation was examined in the paper by Bush, Unal, Rowell and Rehder [98] who used a Taguchi metnod to optimize the weight of a lunar transfer vehicle which implemented aerobraking. The paper, even though not involving RSM attempted to use the DOE theory, through the Taguchi "cook-book" approach, due to the discrete nature of some of the six design variables. The paper acknowledges that the use of the Taguchi matrices

reduced significantly the number of experimental configurations required to identify the design parameters. The paper also suggests coupling the Taguchi method with finite element analysis as an effective approach for conceptual-preliminary level aerobrake optimization studies.

An overview of RSM modeling for first and second order surfaces is presented by Cavanos [29]. The report brings together the basic concepts of RSM. Even though the report does not go beyond a simple presentation of a textbook example for three variables using a CCD method, it emphasizes that the methods presented are crucial for well-planned experimental test. It recognizes that the techniques described are not only essentially optimal in the sense of yielding prediction equations which minimize the error between a measured and a predicted response but are also economical in that they require a relatively small number of measurements.

Finally, Engelund, Stanley, Lepsch, and McMillan [52] present the first complete analysis of a simple aerodynamic configuration design using RSM. Their work described the optimization of six design variables for the preliminary configuration design for a single-stage-to-orbit re-entry vehicle. The authors used once more a CCD method, which produced a second order response surface. They then located the minimum on this surface. The results demonstrated adequately the efficiency of the method, especially when compared with a full factorial design. Unfortunately, the report did not demonstrate the possibility for a sensitivity analysis based on the predicted response surface. Also the authors failed to present the information that was obtained from the Analysis of Goodness they performed on the response surface approximation, very important for the early prediction of the main factors' effect on the objective, and also for the early identification of any interactions between the factors.

In general, the attempts to implement RSM in the aeronautical discipline are limited to simple problems intended for the demonstration of the principle

rather than the solution of real-life problems.

Chapter 3

Design of Experiments and Response Surface Methodology Formulation

The method of Design of Experiments introduces methods that only require a small number of simulations to describe the design space. Some of these methods define specific sequences of point designs to be evaluated, which then are used to fit a second order surface, called a Response Surface. These designs are called second order designs. Such designs only containing the minimum number of points (for n variables the minimum number of points required to determine the coefficients of the second order surface is $1/2(n+1)(n+2)$) are called minimum-point designs.

In cases where the resources are limited or when the cost of the experimental runs is prohibitively high, these minimum-point second order designs are attractive. Central Composite Designs (CCD) are traditionally implemented in such cases, while designs derived from regular polyhedra are sometimes also implemented.

These methods possess characteristics that are important, such as the mu-

tual orthogonality of the coefficients, except for the constant and quadratic terms. This property can easily help in improving of the typical response models. Another important property these designs have is rotatability which assures that the design space is to be explored in an ordered fashion, and generally accelerates the convergence to the optimum region.

3.1 Hybrid Methods

Other near minimum-point designs were also created that under specific assumptions and conditions exhibit more attractive properties than those of the CCD. One type of these designs is the Hybrid Designs, introduced by Roquemore [137]. In his paper, Roquemore describes in detail his Hybrid designs and their properties. In as mentioned before, the most attractive properties of the designs are their orthogonality, near-saturated, near-rotatable and minimum point in size.

Roquemore in his paper presents three designs for three, four, and six variables with 10,15, and 28 points respectively. The proposed Hybrid design for five variables would produce 25 points, only marginally smaller than the existing CCD for five variables.

Roquemore's table demonstrating the combination of factors for six factors is shown in Table 3.1. The Design below is named Hybrid 628A, denoting six variables and twenty eight runs. The A denotes the first possible arrangement. In this case, there are two such arrangements complementary to each other.

3.2 Minimum Point Designs

For more than six variables there is no known method that can guarantee minimum point designs. In this case, as mentioned, it is still desirable to begin

Design 628A					
x_1	x_2	x_3	x_4	x_5	x_6
0	0	0	0	0	$4/\sqrt{3}$
-1	-1	-1	-1	-1	$1/\sqrt{3}$
1	1	-1	-1	-1	$1/\sqrt{3}$
1	-1	1	-1	-1	$1/\sqrt{3}$
-1	1	1	-1	-1	$1/\sqrt{3}$
1	-1	-1	1	-1	$1/\sqrt{3}$
-1	1	-1	1	-1	$1/\sqrt{3}$
-1	-1	1	1	-1	$1/\sqrt{3}$
1	1	1	1	-1	$1/\sqrt{3}$
1	-1	-1	-1	1	$1/\sqrt{3}$
-1	1	-1	-1	1	$1/\sqrt{3}$
-1	-1	1	-1	1	$1/\sqrt{3}$
1	1	1	-1	1	$1/\sqrt{3}$
-1	-1	-1	1	1	$1/\sqrt{3}$
1	1	-1	1	1	$1/\sqrt{3}$
1	-1	1	1	1	$1/\sqrt{3}$
-1	1	1	1	1	$1/\sqrt{3}$
2	0	0	0	0	$-2/\sqrt{3}$
-2	0	0	0	0	$-2/\sqrt{3}$
0	2	0	0	0	$-2/\sqrt{3}$
0	-2	0	0	0	$-2/\sqrt{3}$
0	0	2	0	0	$-2/\sqrt{3}$
0	0	-2	0	0	$-2/\sqrt{3}$
0	0	0	2	0	$-2/\sqrt{3}$
0	0	0	-2	0	$-2/\sqrt{3}$
0	0	0	0	2	$-2/\sqrt{3}$
0	0	0	0	-2	$-2/\sqrt{3}$
0	0	0	0	0	0

Table 3.1: Hybrid 628A Design Table

by fitting,by least squares, a second order model of the form:

$$y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \sum_{j=1}^{k} \beta_{i,j} x_i x_j + \epsilon \tag{3.1}$$

to approximate the response variable y on k scaled predictor variables. It is further assumed in this case as was before that the variables are scaled so that the design points lie in or on the unit cube $-1 \leq x_i \leq 1, i = 1, 2, \ldots, k$.

As minimum point designs, the designs must have $1/2(k+1)(k+2)$ points while at the same time the design should give rise to least squares estimates with minimum generalized variance. To achieve the latter, the determinant $|\mathbf{X}'\mathbf{X}|$ [1] should be as large as possible.

However, no such designs are known for $k \geq 5$ (the six variables hybrid design has near-rotatability) Box and Draper[23] suggest a method producing minimum-point designs without enforcing rigorously the $|\mathbf{X}'\mathbf{X}|$ criterion. In cases where a few more experimental runs are not as computationally expensive to obtain, additional checks on the fit of the response surface can be made. The method by Box and Draper is the one suggested for designs with more than six variables, since the Central Composite Designs produces a prohibitive large amount of runs.

3.2.1 Selection of Design

The choice for a particular hybrid design will be derived from the accuracy requirements of the response surface and also from the imposed requirements in terms of number of variables, cost of each point evaluation. To satisfy the goodness of fit requirements, the hybrid designs have 1 degree of freedom for estimation of fit if the experimenter includes a center point[137].

---

[1] $\mathbf{X}$ is the $n \times n$ matrix whose u-th row is contains the following elements: $(1, x_{1u}, x_{2u}, \ldots, x_{ku}, x_{1u}^2, x_{2u}^2, \ldots, x_{1u}x_{2u}, \ldots, x_{k-1,u}x_{ku})$

In cases where no hybrid designs exist, or they are not more efficient either that Central Composite Designs or other designs, the method of minimum-point near-optimum | X'X | criterion proposed by Box and Hunter can be used.

3.3 The System Model Setup

The main reason that the Design of Experiments and Response Surface Methodology were chosen to create a System Model is that through a minimum number of simulations, it can be possible to draw general conclusions on the behaviour of the response, (i.e. the objective function) and the interactions between the independent variables.

The System Model consists of a second order (quadratic) hypersurface in the n-dimensional space spanned by the n independent variables that enter the System. This hypersurface is constrained by the various physical or multidisciplinary constraints, such as maximum span of fillet, minimum volume allocated to the fillet, maximum leading edge fillet elongation at the wing-root, etc. These constraints are transformed into corresponding constraint hypersurfaces, and the problem becomes one of non-linear optimization with generally non-linear constraints.

3.3.1 RSM Optimization

The second order response surface is analyzed and the minimum point on the surface determined by using a classical optimizer with second derivative numerical evaluation. The reason an optimizer had to be implemented in this case where the stationary point can be analytically determined by basic multivariable calculus is that this is a case of constraint optimization, where this particular stationary point may lie outside of the feasible region. In such a

case, it holds no particular significance and the information it provides can be ignored.

3.3.1.1 Theoretical Considerations

The response surface methodology is a case of nonlinear multiple regression [50], implying that the variables appearing in the regression equation may have a degree higher than unity, or they may be cross-products, or *interactions.* Since in the Response Surface formulation (see Eq. 3.1), these nonlinear terms enter into the equation in an additive manner. This eliminates computational complications, since by substituting a higher order term or an interaction term as simply an extra variable, the equation can be derived in exactly the same way as is described in the statistics literature (See [50],[95],[148]).

In addition, by the inclusion of the interaction terms, the second order surface provides information on the relative importance of the independent variables, similar to a Resolution IV [44] experimental design, only with fewer objective evaluations required.

3.3.1.2 Program Description

The user specifies the number of variables, the variable value bounds (upper and lower) and the appropriate design table (e.g. Table 3.1 is selected (Box's minimum-point or hybrid minimum-point). The variables are then normalized according to the design table extremes. Finally, the response surface is obtained for each line of the design. The process is outlined in Fig 3.1

The variable normalization according to the Design of Experiments Table consists of producing the variable values required for the Table of Experiments from their natural lower and upper limits. Assume that one of the six design variables that will enter the Table of Experiments described in Table 3.1 ranges between 2 and 50. Furthermore, assume that this variable will be the variable

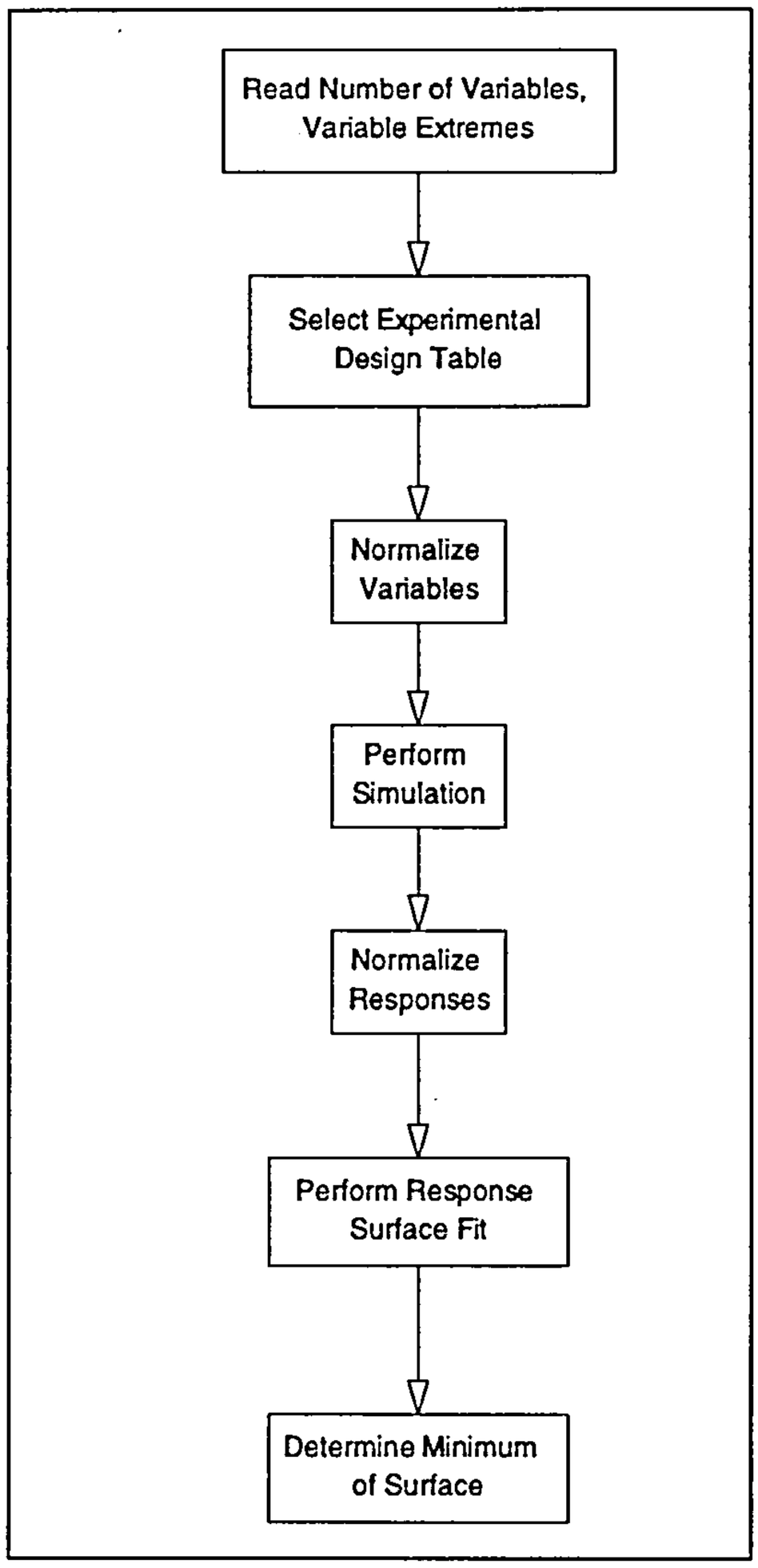

Figure 3.1: Design of Experiments Table and Response Surface Setup

	Min	*Max*					
DOE Table Values	-2	2	-2	-1	0	1	2
Variable 1 Values	2	50	2	14	26	38	50

Table 3.2: Producing Variable Values for the Experiments Table

described in the first column of Table 3.1 which contains five different levels of the variable, namely $-2, -1, 0, 1, 2$. Table 3.2 describes this process while Table 3.3 contains the entire table of values for the six variable simulation performed:

The input variables and their ranges are defined and described in Chapter 6.

The transformation of the Experiment Table variable levels is followed by the execution of the program which provides the response of the system to the input variables at the levels defined. The program structure is described in Appendix F. After the responses are obtained, they are transformed for later use in the surface fit. The general non-linear regression transformation of the response is: $Y' = Y - \overline{Y}$, where Y' is the transformed response, Y is the actual system response, and $\overline{Y}$ is the mean of the system responses.

The transformed responses are then used to produce a second order surface (Eq. 3.1) using non-linear regression. The variable transformations are shown in Table 3.4 where $\overline{x}_i^2$ is the mean of x_i^2, i.e. the i-th quadratic term mean.

3.3.2 Relevant Work on Similar Problems

The choice of a second-order surface to predict the response of such a complex system may initially seem arbitrary. Nevertheless, after consulting with a number of industries in the field, and receiving information and/or advice on their methods of evaluating interference drag in wing-fuselage junctions, the

Design 628A					
Factor 1	Factor 2	Factor 3	Scale X	Scale Z	Span
26	26	101	1.55	2.05	6.500
14	14	52	1.325	1.525	5.152
38	38	52	1.325	1.525	5.152
38	14	153	1.325	1.525	5.152
14	38	153	1.325	1.525	5.152
38	14	52	1.775	1.525	5.152
14	38	52	1.775	1.525	5.152
14	14	153	1.775	1.525	5.152
38	38	153	1.775	1.525	5.152
38	14	52	1.325	2.525	5.152
14	38	52	1.325	2.525	5.152
14	14	153	1.325	2.525	5.152
38	38	153	1.325	2.525	5.152
14	14	52	1.775	2.525	5.152
38	38	52	1.775	2.525	5.152
38	14	153	1.775	2.525	5.152
14	38	153	1.775	2.525	5.152
50	26	101	1.55	2.05	3.500
2	26	101	1.55	2.05	3.500
26	50	101	1.55	2.05	3.500
26	2	101	1.55	2.05	3.500
26	26	200	1.55	2.05	3.500
26	26	2	1.55	2.05	3.500
26	26	101	2.00	2.05	3.500
26	26	101	1.1	2.05	3.500
26	26	101	1.55	3.00	3.500
26	26	101	1.55	1.1	3.500
26	26	101	1.55	2.05	4.703

Table 3.3: Hybrid 628A Transformed Value Table

Transformed Variable	*Original Variable*	
z_i	x_i	first order terms
z_j	$x_{i_1} x_{i_2}$	interaction terms
z_k	$x_i^2 - \overline{x}_i^2$	second order terms

Table 3.4: Nonlinear Regression Variable Transformations

following became apparent:

In many fillet optimization attempts, often a modified Simplex method produced reasonable results that where later confirmed with wind-tunnel tests[2]. This method was also based on surface optimization of a surface definition based also on Bezier-Bernstein parameters, and an objective function which consists of a combination of a modified Full Potential code and a semi-inverse boundary layer code. Other manufacturers have stated that most of their fillet optimization process is performed in the wind-tunnels, with models of fillets being adapted onto the aircraft models and measurements then are taken. Their results indicate that by inter/extrapolating between the various fillet parameters (i.e. radius of curvature of the fillet, length of the leading edge portion, length of the trailing edge portion) they could produce fairly predictable results for small variations of these properties[3]. This apparent superposition of behaviours suggests that at least in the area of interest in the feasible region, a fairly linear response is observed. From the above, it was reasonable to choose a quadratic surface to represent the response of the system over a wider region of interest.

3.4 Method Overview

The implementation of Design of Experiments and Response Surface Methods to model a complicated system such as the force system acting on the wing-fuselage fillet presents three main advantages. First, the number of simulations required is significantly low, when minimum-point designs are used. Second, the quadratic response surface provides relatively accurate predictions over the confidence intervals it is applied on, and finally, due to the Experimental Designs that are near rotatable (the hybrid tables as well as the experimental

[2]From personal correspondence with Short Brothers plc.
[3]From personal correspondence with Aerospatiale and Airbus

tables provided by Box and Draper [23] being based on the CCD designs) the variance of the response is independent of the sampling of the design space, i.e. the choice of the independent variable values.

The most important drawback of the method is its inability to correctly predict non-smooth objective functions with responses that differ significantly from those of a second-order surface. It is essential, therefore, to ensure before the simulations occur, that the objective function exhibits relatively smooth behaviour in the region of interest, and that a second-order response surface will be adequate to predict the response of the system in any point within the region of interest.

Chapter 4

CFD Modelling

4.1 Introduction

At the wing-fuselage junction, the boundary layer of the fuselage surface, as it approaches the wing leading edge exhibits adverse pressure gradients because of the presence of the wing in the flow [99]. There follows a potential separation of the boundary layer ahead of the wing leading edge and the formation of a horse-shoe vortex which wraps around the wing and extends further downstream [6][7]. These two effects, namely the horseshoe vortex along with the usual secondary flows present in any streamwise corners lead to a complicated three-dimensional flow field around the juncture [87]. This flow is unstable with the vortex attenuating and even disappearing at times. This complicated flow is further intensified by the effects of flow separation near the wing's trailing edge [126]. The flow around the junction thus usually creates incremental drag, while the horse-shoe vortex can strongly influence the lift and the aerodynamic behaviour of the wing-root by modifying the regions of attached flow. To analyze such a flow, the EAGLE Thin-Layer Navier-Stokes solver was chosen since this was the only Navier Stokes Solver available to Cranfield University at the time. Use of full potential flow solvers or even

Euler solvers was considered inadequate for this problem since the viscous three-dimensional behavior of the flow field around the junction would require viscosity and turbulence modeling. Nevertheless, a comparison of the results obtained from the Thin-Layer Navier-Stokes code and an Euler code may be of interest. Such a comparison though is beyond the scope of this study.

4.2 Geometry Modeling

The aircraft geometry was obtained from Cranfield University's College of Aeronautics (Aerospace Vehicle Design Group) group design project for 1995, a large capacity commercial transport [130]. The aircraft was modeled in Unigraphics, and sections of the fuselage and the wing were provided as inputs for the geometry definition in this thesis.

4.2.1 Fillet Description

The wing-fuselage fillet was defined as a pair of nonuniform Bezier surfaces, with control points defining a net which, in turn, defined the shape of the surface. The advantage of such a formulation lies in the ease of modifying the surface either locally or globally. The Bezier net and surface exhibit the property that the latter always lies in the convex hull of the former [53], which, combined with the variation diminishing property of Bezier surfaces provides a reliable, predictable and efficient way of defining a surface.

Bezier curves are the most widely used curves that provide an efficient way of handling geometrical descriptions. In terms of the least number of points required, the Bezier curves are an obvious choice when compared to B-Splines. Although B-Splines and B-Patches provide a greater flexibility in general for arbitrary curve approximation, the Bezier curves require much fewer parameters. The latter implies that for a surface optimization scheme, fewer

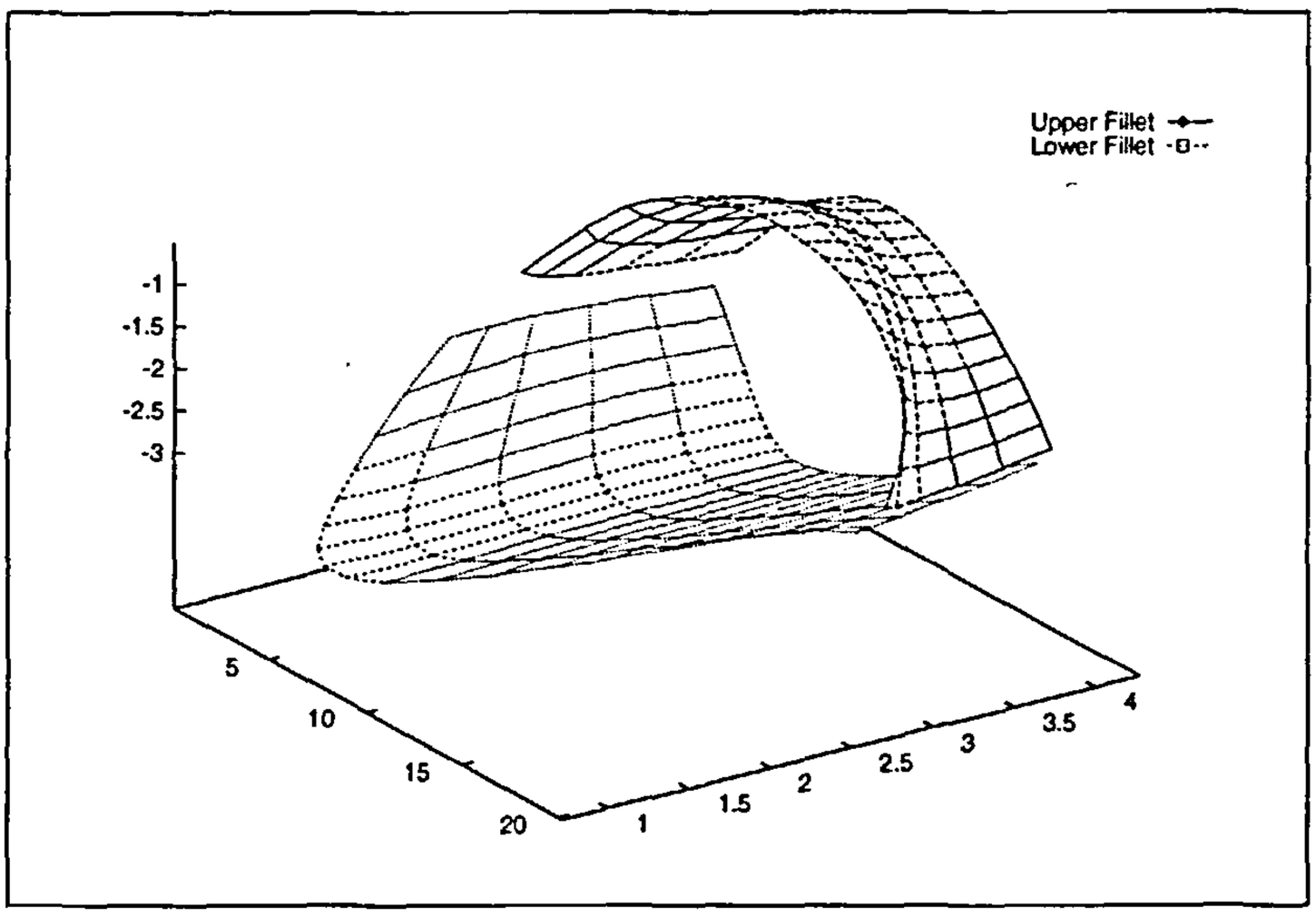

Figure 4.1: Example of Fillet Definition (Surfaces are shrunk for clarity)

input variables (the control net points of either Bezier of B-Spline surfaces) could imply a vital reduction in iterations. In cases where computational efficiency and cost are important, Bezier curves are the natural choice.

4.2.2 Implementation

The fillet is defined by a pair of nonuniform 5x4 Bezier nets, which take into account the local derivative information at the bounding curves and provide tangency with both the fuselage and the wing at the intersection points. The tangency requirement is not rigidly enforced, and small deviations are allowed in the fuselage-fillet junction, (See Figure 4.1 where the top fillet surface is allowed not to fully satisfy the tangency to the fuselage surface requirement at the joining curve of the fillet surface with the fuselage surface.)

The lower surface fillet is free to extend over the fuselage and cover an area as much as required by the user/optimizer input (See Figure 4.2). Furthermore, the location of the intesection with the upper and lower fuselage may

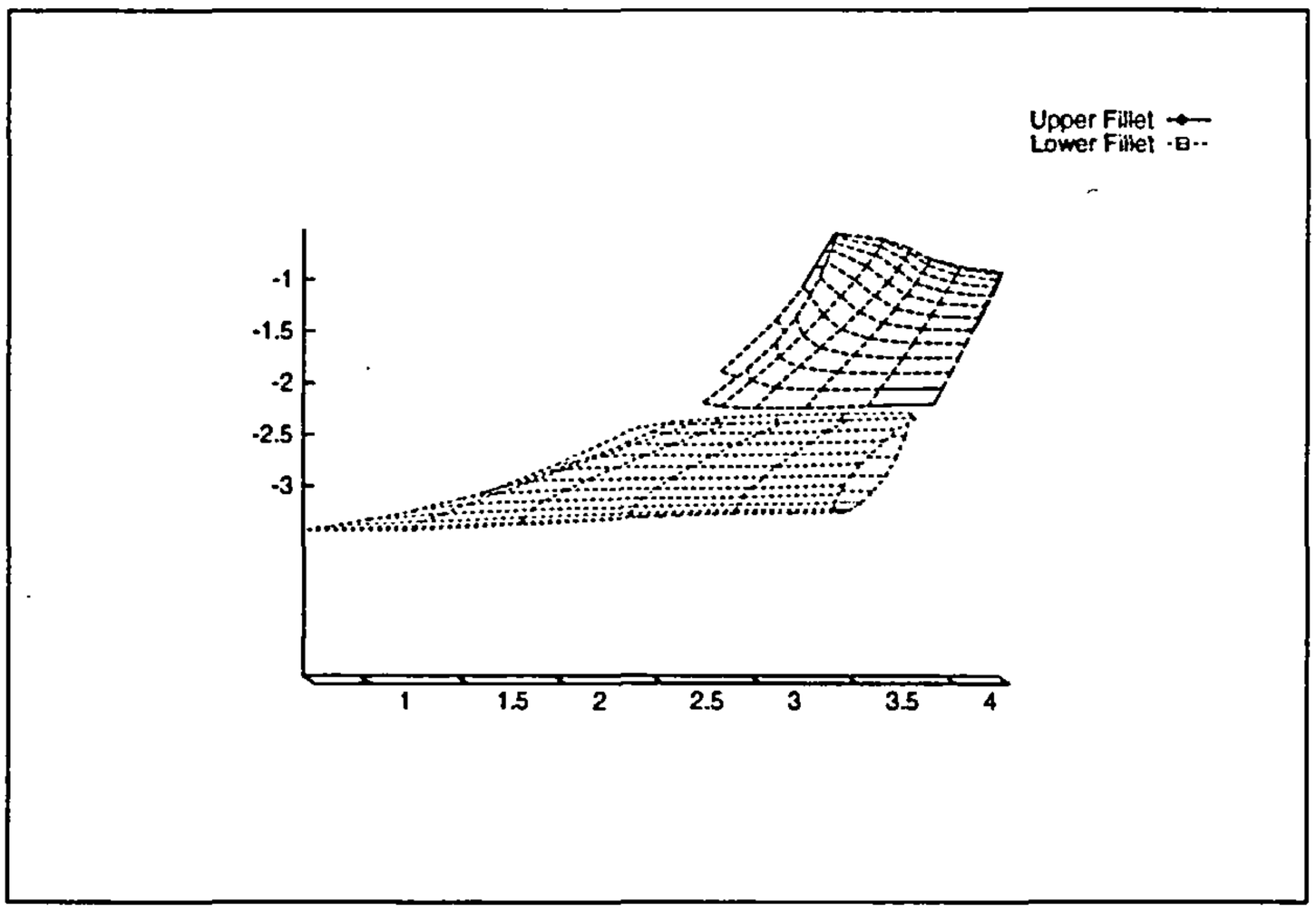

Figure 4.2: Example of Fillet Definition (Surfaces are shrunk for clarity)

be changed, in effect move higher in the upper fuselage and lower in the lower fuselage, or move ahead of the wing-root section leading edge and further behind the wing-root section trailing edge, introducing thus a leading edge and a trailing edge fairing. This is achieved by a scaling of the wing-root section and by a subsequent translation to the location of interest. This is done separately for the upper and lower fillet, although care is taken to ensure that the leading and trailing edges of the upper fillet-root section coincide with the corresponding leading and trailing edges of the lower fillet-root section (See Figure 4.3)

Finally the fillet is free to "expand" or "contract" by varying the length down the span of the wing at which the fillet-wing intersection will occur. This is controlled by the user by specifying the limiting value of the fillet expansion based on practical matters (i.e. flap operation). The fillet-wing intersection locations for both the upper surface and the lower surface fillet are identical.

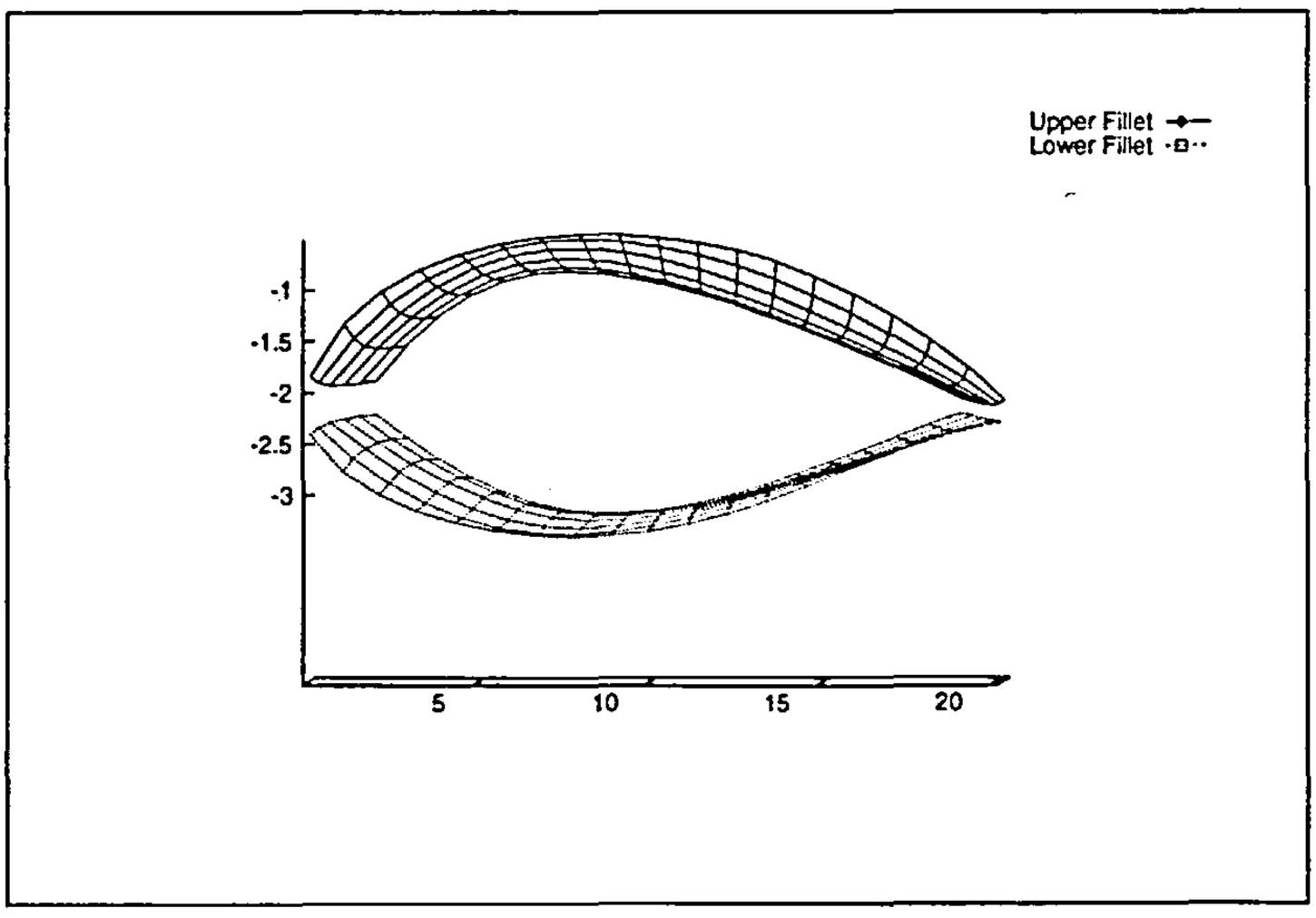

Figure 4.3: Example of Fillet Definition (Surfaces are shrunk for clarity)

4.2.3 Geometry Definition

The surface of the aircraft and the fillet were defined using the Surface Generation Module of the EAGLE package[61]. The operation and structure of the module is described in Appendix A while the process of defining, constructing and integrating the fillet surfaces into the code are outlined in Appendix F. The geometry was defined in terms of a text file, since no appropriate graphical interface was available.

Since the aircraft was analyzed for cruise conditions, only half the aircraft was modeled due to symmetry (See Figure 4.4. It was not possible to obtain symmetry conditions for the upper and lower surface of the aircraft as well, due to the dihedral. Therefore, the surface of the fuselage and the wing were divided into upper and lower fuselage and upper and lower wing surface respectively (See Figure 4.5). Two computational blocks were created, one encompassing the upper surface of the fuselage, the upper wing surface and part of the flowfield, while the second computational block contained the remaining

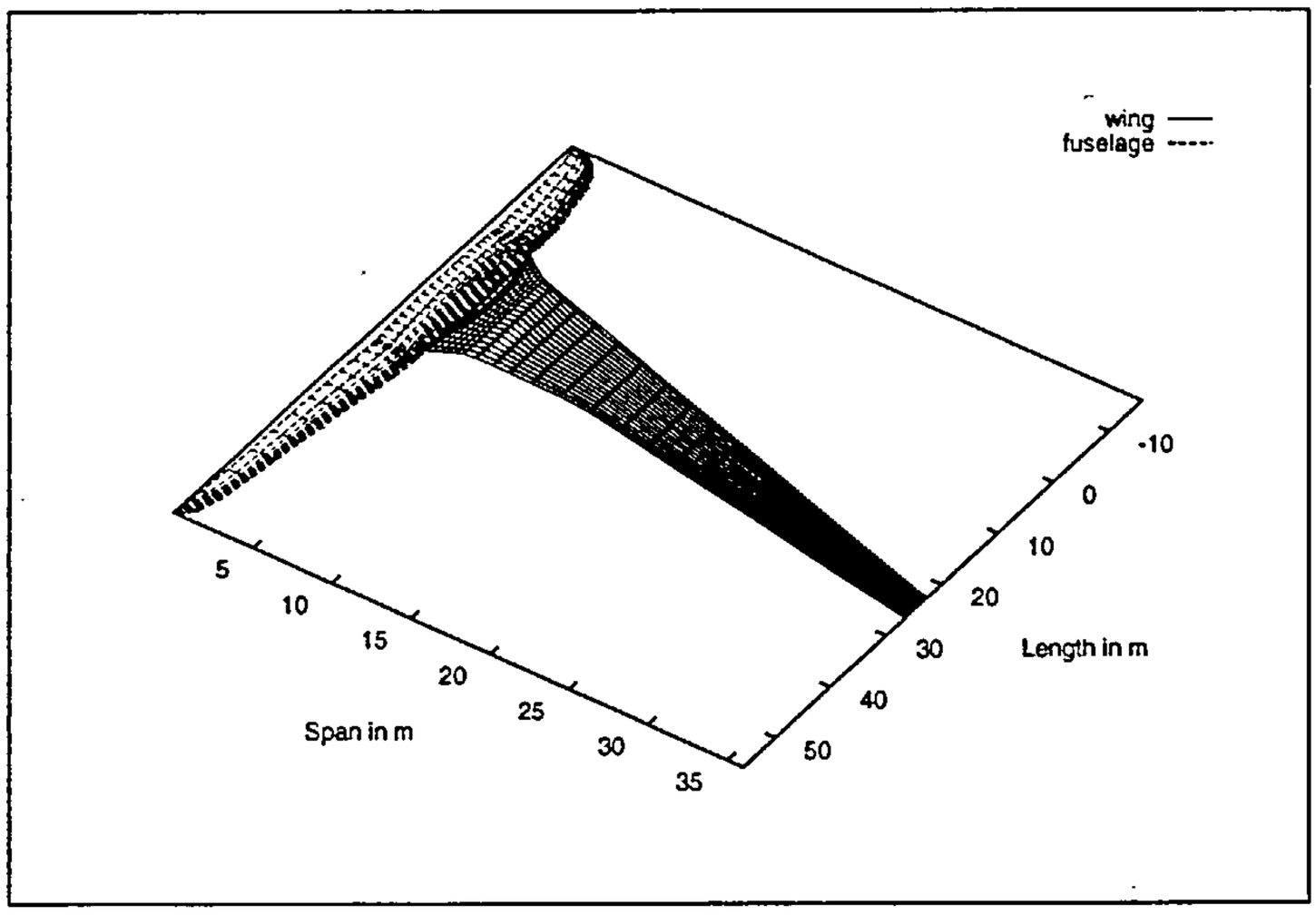

Figure 4.4: Half Body Aircraft Model

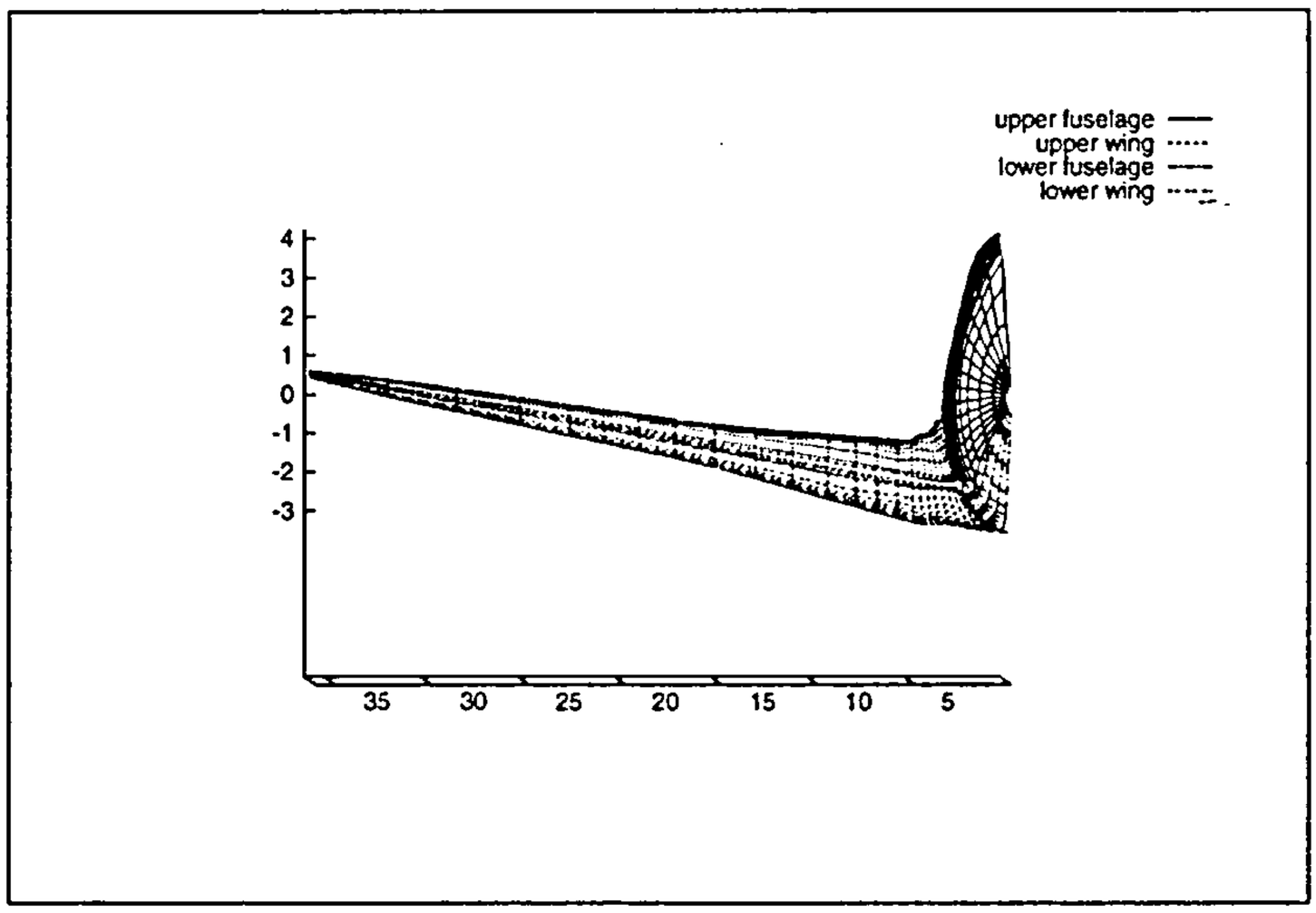

Figure 4.5: Aircraft Surface Definition

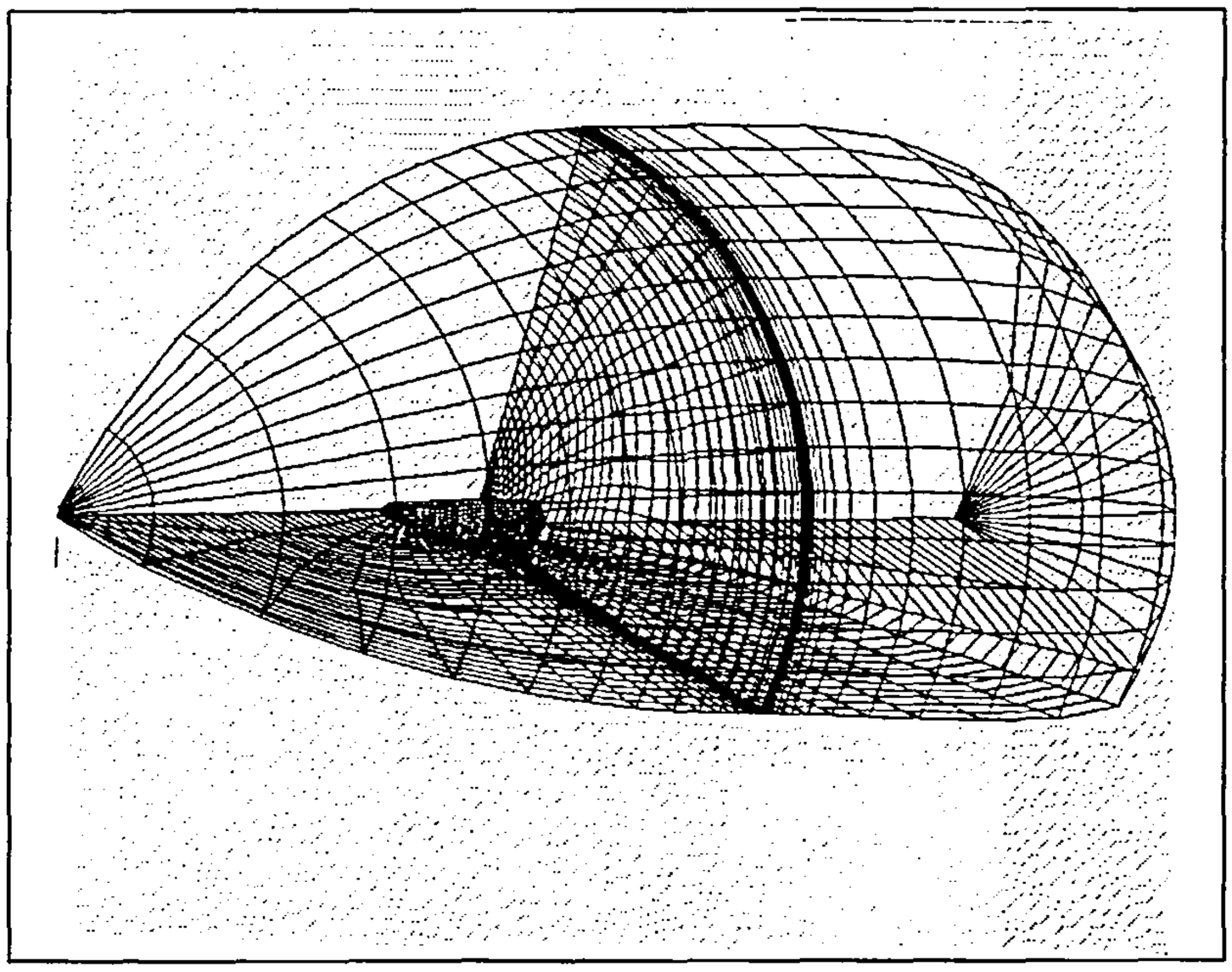

Figure 4.6: First (Upper) Computational Block

part of the fuselage and the lower wing surface (See Figures 4.6 and 4.7)

4.3 Grid Generation

An O-type grid was generated automatically by the EAGLE package Grid Generation Module. The input format and the options available are described in Appendix B. Once more, there was no graphical interface available for the creation and modification of the geometrical properties of the grid. The size of the O-type grid was $55 \times 35 \times 20$ for both blocks, for the half-body, with a resolution of 20 points on each airfoil section, and 25 airfoil sections for the wing, densely packed closer to the junction and with a larger spacing further down the span. The grid resolution was initially $70 \times 60 \times 60$ but the processing time required for the runs on the Cranfield University Cray was prohibitively high (20-26 hours a run). Therefore, the grid became coarser. Similar size

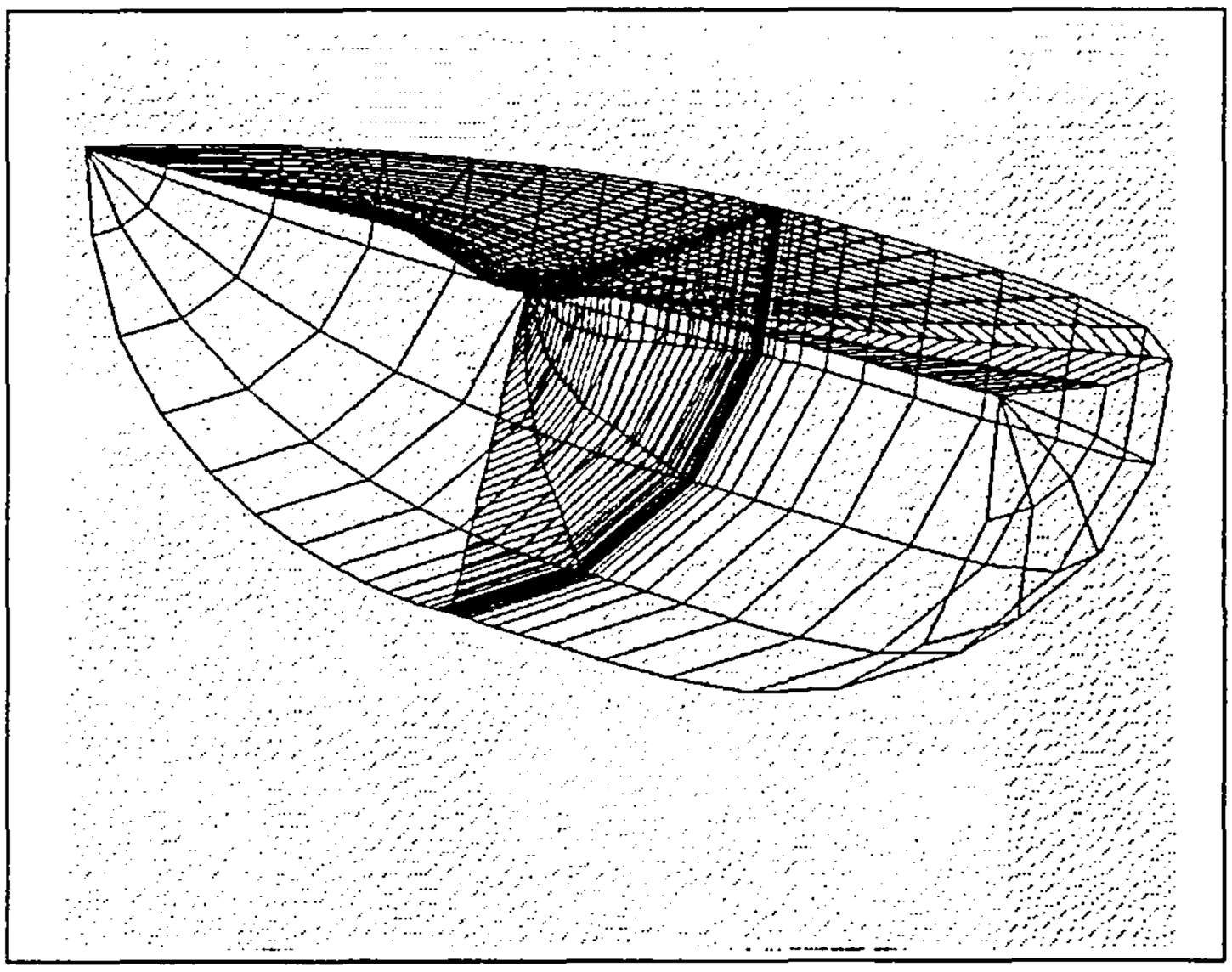

Figure 4.7: Second (Lower) Computational Block

grids have been used to demonstrate junction flow features, and also by EAGLE practicioners to simulate flows around vehicles [104][158]. It is however common practice to use much larger grids (of the order of 10^6 points) to evaluate full aircraft configurations in viscous transonic flows. The final elliptic grid was based on an initial algebraic grid which was then iterated upon with appropriate control functions and after a number of iterations the final grid was produced. It was determined that for the particular geometry and requirements of this case, twenty iterations were sufficient for a grid that was dense in the junction area, orthogonal to the aircraft surfaces, with one computational block surface subject to symmetry conditions and another classified as inter-block surface (i.e. a surface common to two computational blocks.) The description of the computational blocks and their surfaces appears in Figure 4.8. The term "leading edge singularity" refers to the singularity line which connects the frontmost point of the fuselage with the furthest point of the far

field grid boundary upstream. Similarly, the term "trailing edge singularity" refers to the singularity line connecting the rearmost point of the fuselage with the furthest point of the far field grid boundary downstream. In the same Figure the common surface in both blocks is denoted along with the vertices that coincide in the physical space. The two surfaces connecting the fuselage surface with the outer *far-field conditions* surface had to be collapsed into two singularity lines in physical space, but they maintain their surface properties in the computational space. Finally, the far field boundary was placed at 15 wing–root chords upstream from the fuselage, 30 wing–root chords downstream from the rearmost fuselage section, and 2 wingspans away from the wingtip.

4.4 Thin–Layer Navier–Stokes Solver

The EAGLE package provides a Thin-Layer Navier-Stokes code which accepts the formatted grid file from the Grid Generation module and performs the flow analysis. The input description and specifications are described in Appendix D. The code is vectorized for the Cray YMP, and it runs relatively fast on Cranfield's J19. For the $55 \times 35 \times 20$ grid, the code would require approximately one minute per iteration, depending on the options specified in the input code (See Appendix D). Original runs took considerably long time since the approximate number of iterations for convergence had to be determined. Once this figure (between 900-1300) was determined, an average run would require between ten and fifteen hours to complete. The convergence was determined mainly by the consistency of the results and a typical history of the residuals over the number of iterations is shown shown in Figure 4.9 (where one point every twenty iterations is shown):

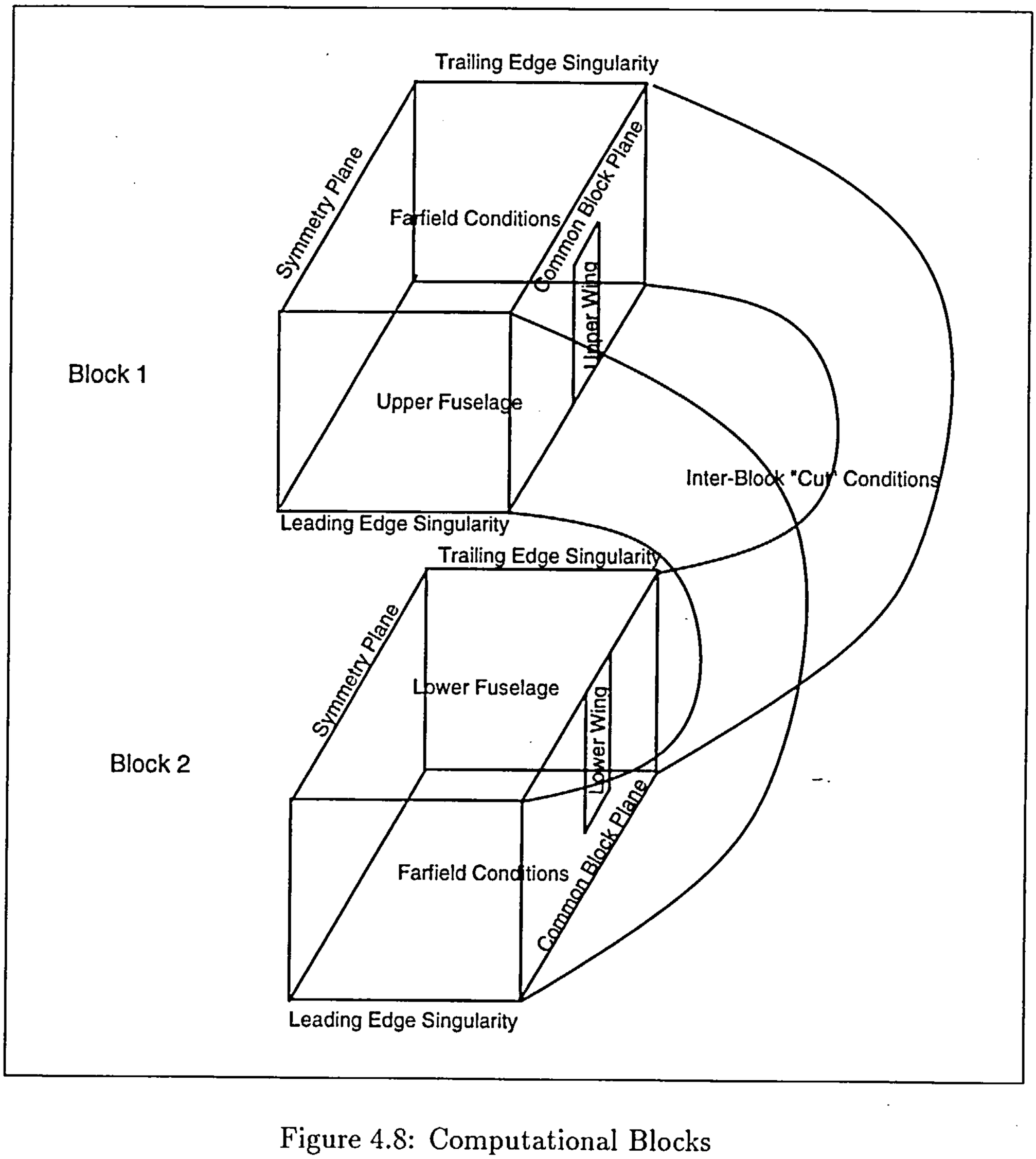

Figure 4.8: Computational Blocks

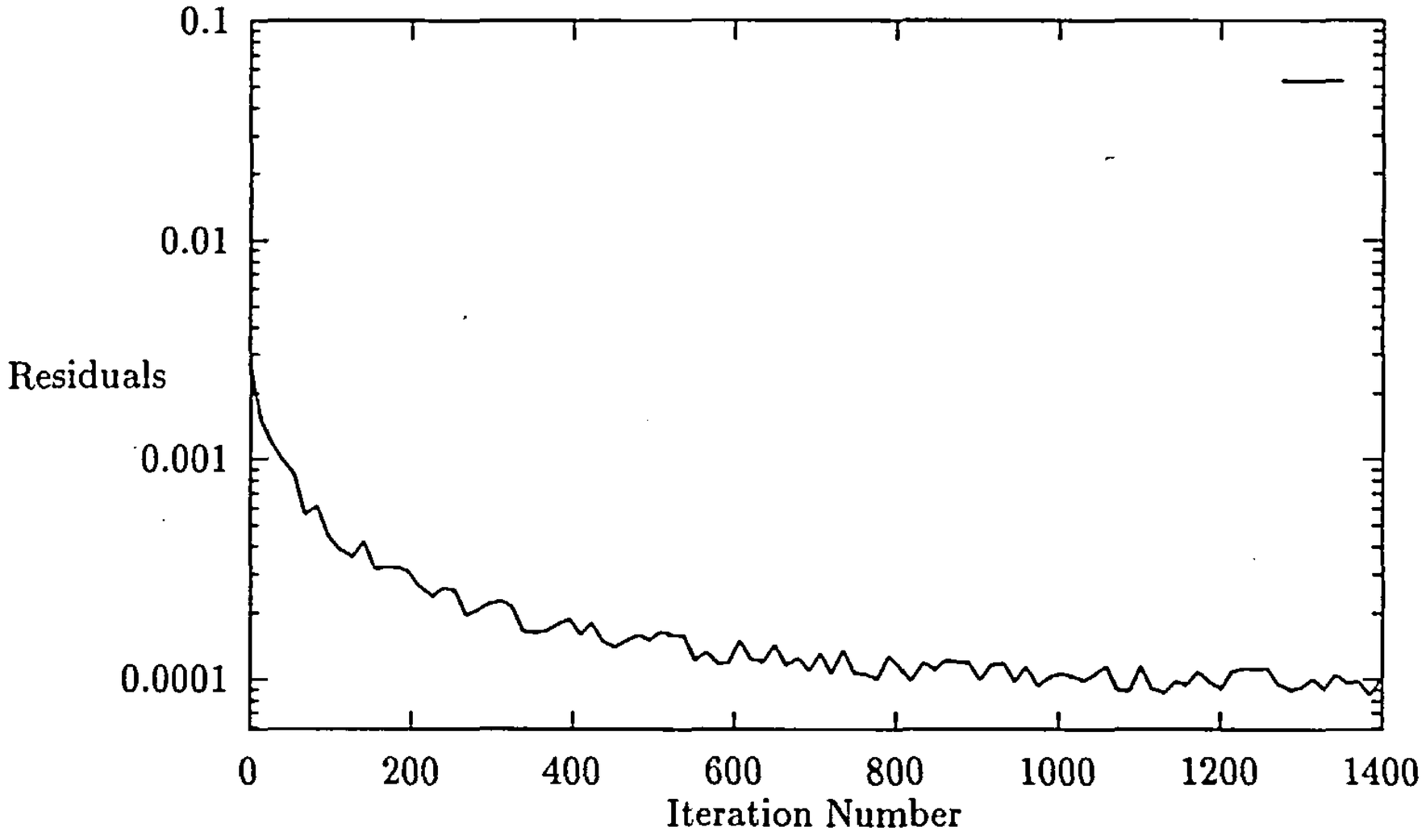

Figure 4.9: Example of Convergence History

4.4.1 The EAGLE TLNS Solver

The following provide the framework in which the simulations presented in this thesis took place. It describes the flow conditions examined, the model used, the grid generated, and the boundary conditions imposed on it.

4.4.1.1 Flow Conditions

The original assumption made for the conditions of the flow to be used for the simulation was that the aircraft was at cruise conditions, at Mach 0.85.

4.4.1.2 The Model

The declarations in the flow solver input file present the main assumptions made for the mode. The CFL (See Appendix D) number was chosen to be 7.00 and it was thought to represent adequately the angle of attack, the bluntness of the configuration and the Mach number after the suggestion of the EAGLE

developers. The lift axis was defined as the positive z-axis. This implies that the solver resolves the lift into a z-axis componont and a drag component. This, in its turn, implies that any lift changes resulting from the fillet shape will affect the drag. For optimization purposes, lift was not kept constant. This eliminated the need for unsaturated designs that would be necessary to ensure that the design points chosen for the DOE application would all lie within the feasible region.

The Mach number was declared at .85 and the angle of attack was set at 4 degrees. Based on the wing-root chord, the Reynolds number used was 8.5×10^7.

The frequency of updates to the flux Jacobians was set to 10. It was found that this was a good compromise between the time required per iteration and the total number of iterations for convergence. The splitting technique employed was the flux-difference-splitting of Roe, since the Steger flux-vector splitting was not fully implemented in the code. The turbulence model used in the code is the Baldwin-Lomax model since this is the only model available in the solver.

The number of zero pressure gradient iterations was set to 75, which is a reasonable value for 3D configurations at these Mach numbers, taking into account the bluntness ratio of the configuration and the angle of attack.

4.4.1.3 The Grid

The final elliptic grid that was used as input for the flow solver consisted of $55 \times 35 \times 20$ points. It was created in two blocks, the simplest arrangement, given the flow solver capabilities, to represent adequately the geometry of the aircraft. The upper block contained the upper portion of the fuselage and the upper surface of the wing (See Figures 4.8,4.6) while the lower block contained the lower portion of the fuselage and the lower surface of the wing

(See Figures 4.8,4.7). The dimensions of the first block were $55 \times 35 \times 15$ while the dimensions of the second block were $55 \times 35 \times 5$. The reason for the small number of the third component of the second block is that it represents meridians along the fuselage. In the second block, the fillet surface is equipped with the capability to almost fully cover the lower fuselage at the junction area (See Figure 4.2. The dense spacing required in the region is even then satisfied with the five meridians running through a very narrow "neck" of surface of the lower fuselage.

4.4.1.4 The Boundary Conditions

Because of the symmetry of the aircraft, only half the body and one wing were modeled. Therefore, two of the surfaces of the two computational blocks (one surface of each block) must be the plane of symmetry (See Figure 4.8). The fuselage surface and the portion of the common block surface that constitutes the wing are classified as solid (fixed) points, while the remaining common surface of the two blocks is classified as inter-block boundary.

To connect the fuselage with the outer boundary of the grid which is characterized as free flow condition boundary two singularity lines/surfaces were created. These are surfaces that were collapsed into a line to create the remaining two surfaces required for the computational block. These two surfaces per block were classified as singularity lines.

4.5 Results

When creating the input file for the EAGLE TLNS solver, the user must define all solid surfaces that experience aerodynamic forces in the flow considered. The first definition of such a *surface* must encompass all the solid (fixed) surfaces of the model. The solver will then produce not only a force and moment

calculation on the overall computational *surface*, but it will also provide the pressure coefficients at every node of the *surface*. If a more detailed force calculation is desired, the user can define additional computational surfaces which will contain only a subset of the model surfaces the original computational surface contained. In the six variable surface optimization demonstration, the drag estimates were obtained by initially comparing the drag of the fuselage in the absence of the wing, the drag of the wing in the absence of the fuselage (with and without a fillet) , and the drag of the wing and fuselage together in the flow, thus obtaining the interference drag for the no-fillet configuration. Further drag estimates were obtained by comparing the full configuration drag (with the fillet) with the drag of the no-fillet configuration, thus obtaining the interference drag change due to the presence of the fillet. Unfortunately, the flow solver is not yet equipped to provide the properties of the flow field at any given grid point. Nevertheless, in addition to an overall force calculation over all the model surfaces, the pressure distribution can also be obtained over the same surfaces, and some conclusions can be drawn. A typical example of the quantitative type of information that is immediately available from the solver is shown in Figure 4.10. Note that the color gradients in the image are artificial and not linearly varying, to facilitate the detection of pressure coefficient gradients.

4.6 Validation

The case considered for initial validation of the EAGLE TLNS flow solver was taken from Sung et al[147]. In the paper, they considered the flow around an airfoil-flat plate junction. In particular, the cases that were considered exhibited the following characteristics:

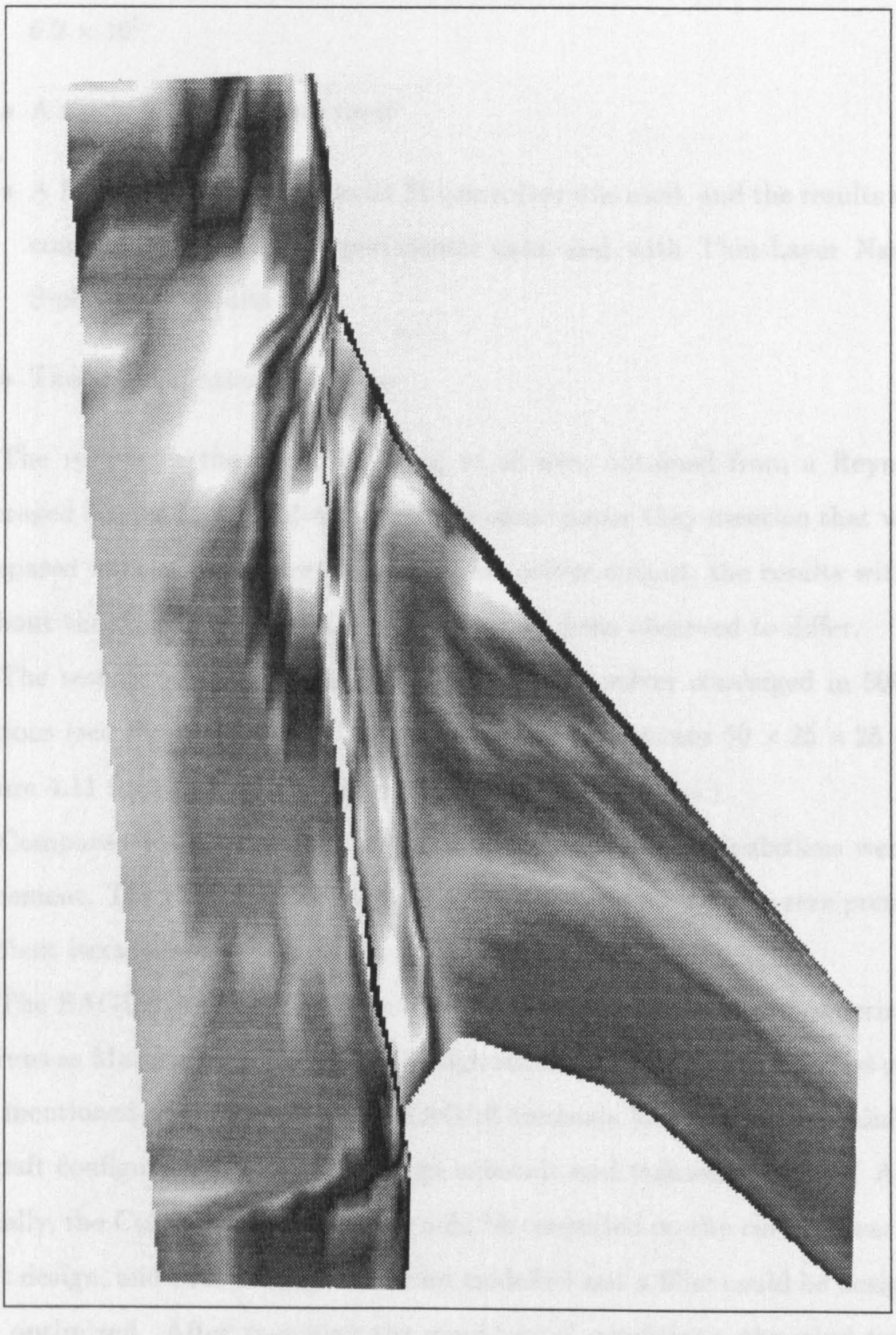

Figure 4.10: Cp Visualization over upper wing and fuselage

- The flow around the NACA 0012 airfoil was at a Reynolds number of 6.2×10^5
- A single block grid was used
- A Reynolds-Averaged Navier Stokes solver was used, and the results were compared both with experimental data and with Thin-Layer Navier-Stokes code results
- The angle of attack was zero

The results in the paper by Sung et al were obtained from a Reynolds Averaged Navier Stokes solver, but in the same paper they mention that when compared with a Thin-Layer Navier-Stokes solver output, the results with or without the thin-layer approximation have not been observed to differ.

The results obtained from the EAGLE TLNS solver converged in 500 iterations (see Figure 4.12 for a C-type grid with dimensions $50 \times 25 \times 25$ (See Figure 4.11 for similar pressure distributions on the airfoil.)

Compared with the results of the above, the EAGLE calculations were in agreement. The CFL number used in this case was also 7 and the zero pressure gradient iterations were set to 60.

The EAGLE program has been also tested against a number of experimental runs as Martinez [104] describes in high subsonic regimes. Further test cases are mentioned as examples in the EAGLE manuals [59] consisting mainly of aircraft configuration runs also in high subsonic and transonic regimes. Additionally, the College of Aeronautics could be consulted on the current year aircraft design, and such a design could be modelled and a fillet could be designed and optimized. After matching the wind-tunnel conditions, the wind-tunnel results could be compared to the solver-optimizer results. For this, it would be interesting to compare the results with an Euler solver and a Navier-Stokes

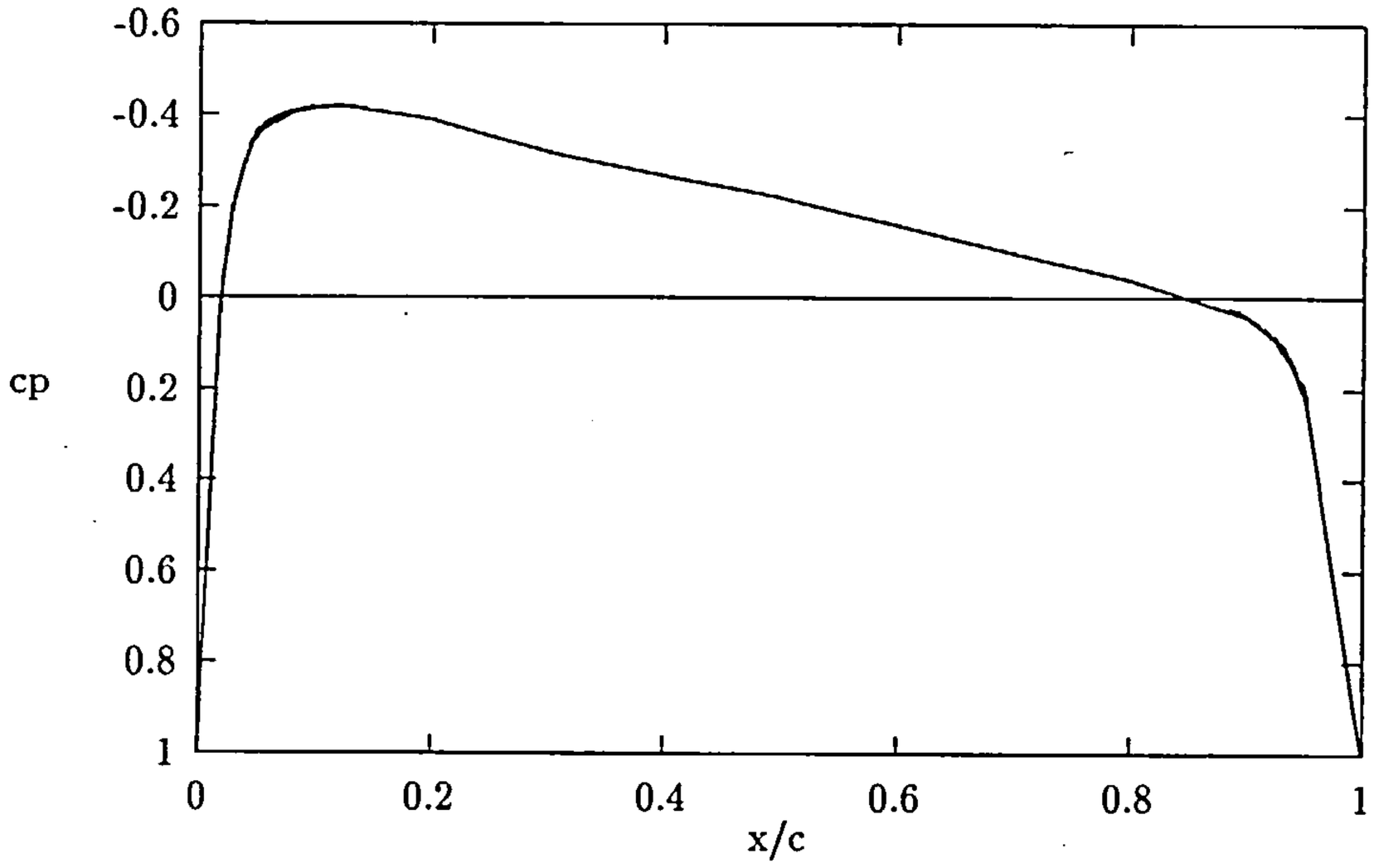

Figure 4.11: Pressure Distribution on NACA 0012 Airfoil, at the Wing-Root, at 0 degrees angle of attack, and $Re = 6.2 \times 10^5$

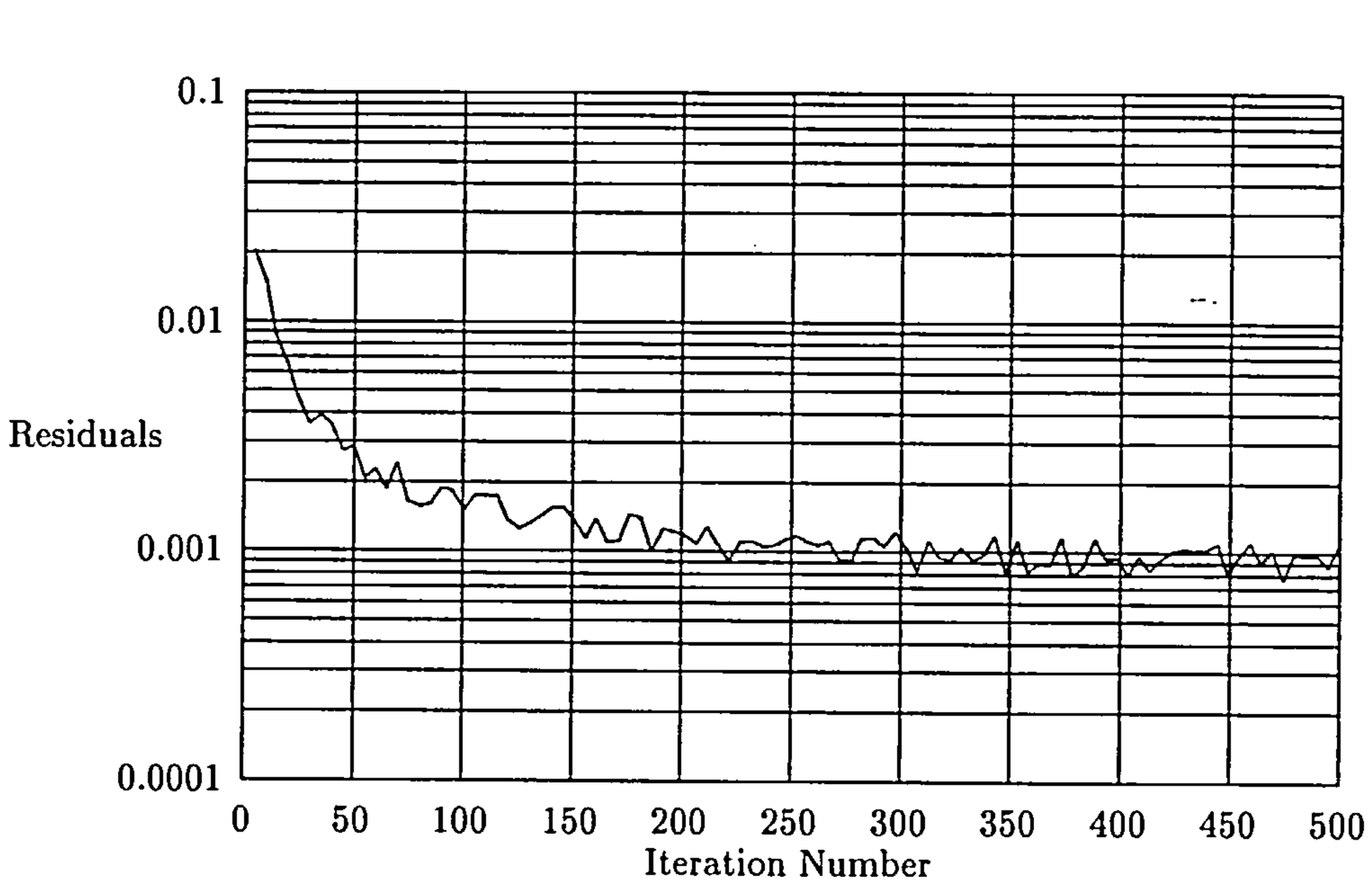

Figure 4.12: Convergence History for the NACA 0012 Test Case

solver to determine whether just an euler code would be adequate to model the flow field at such low Mach and Reynolds numbers. Towards this goal, the low Reynolds Number validation described above could contribute a head start. Towards the same goal, such a venture may additionally expand into different disciplines where junction flows are involved.

4.7 CFD Conclusions

The Thin-Layer approximation is capable of providing adequate results in junction type geometries. With EAGLE, the TLNS flow solver may be run on the Cranfield University's Cray supercomputer, since the code is already vectorized for a Cray machine. Thus it is possible to obtain adequate results in a reasonable amount of time.

The flow solver results, even though appropriate for this implementation, lacked the information content to aid the visualization of the flow field around the junction. It was limited to surface pressure coefficient distributions. The option for a Flow3D data output was not implemented in the version currently available.

Finally, the implementation of the EAGLE package, although capable of handling efficiently the most demanding of geometries and configurations, was heavily lacking in terms of a user interface, since all input had to be carried out in a text editor, in the form of a programming language. A graphical user interface for the three different phases of the program, (i.e. the Geometry module, the Grid module and the Solver module) would greatly facilitate the design of a new geometry and grid and reduce the corresponding time required.

Chapter 5

Landing Gear Analysis Module

5.1 Introduction

The conceptual and preliminary design of an aircraft is a multidisciplinary exercise. It is essential in any design methodology to allow for easy interchangeability of the disciplines, and so to facilitate trade studies that may have to be performed. To demonstrate the modularity of the aerodynamic surface optimization as part of a multidisciplinary process, a module for basic landing gear design and sizing up was incorporated into the optimization loop for the design of the fillet.

5.2 Description of Analysis Method

The purpose of the module is to perform the preliminary sizing of the tire, the initial sizing of the stroke and the leg of the landing gear and finally determine the volume that must be available for the landing gear retraction 5.3. The fuselage-wing fillet is then required to provide at least this volume to accommodate the retracting landing gear. It is assumed that the aircraft will have two main landing gears, each under each wing, and their location is

assumed fixed and determined by the configuration of the aircraft[130].

The module analysis method is based on the preliminary design guidelines provided by Roskam[138] and Currey[39].

5.2.1 Inputs and Outputs

The detailed methodology is described below. The inputs required for the calculation of the required parameters (i.e. Maximum Take-Off Weight of the aircraft) are obtained from the Cranfield University Group Project reports[130]. The output provided by the module, as shown in Figure 5.3, is the volume requirements by the landing gear (See Figure 5.2).

5.3 Landing Gear Analysis

The landing gear sizing module performs a preliminary sizing of the main landing gear for an aircraft and produces the geometrical requirements for the containment of the landing gear within either the wing or the fuselage–wing combination. It is based on the preliminary design guidelines outlined in Roskam [138] and Currey [39].

It is assumed that the aircraft will have two main landing gear units, each under each wing, and that their location is fixed and determined by the configuration of the aircraft. The landing gear unit type is assumed to be dual tandem, as shown in Figure 5.1. The features that are calculated by the module are shown in Figure 5.2. The volume of the block construction is considered to be the storage volume required for the storage of the undeployed main landing gear.

The module performs the preliminary sizing of the tire, the initial sizing of the length of the leg of the landing gear and the stroke, and determines thus the volume that must be available for the retraction of the landing gear. This

process is illustrated in the block diagram of Figure 5.3.

The methodology illustrated in Figure 5.3 is described below:

Given the Maximum Take–Off Weight of the aircraft ($MTOW$) the Maximum Load per Tire can be calculated as follows:

1. The Maximum Ramp Weight (MRW) of the aircraft is 1.005 to 1.01 times the $MTOW$

2. Multiply the MRW by 1.25 [138]

3. Divide this load my the number of tires on each gear to get the design maximum static load per main gear tire ($SLPT$).

To calculate the Maximum tire operating speed, Roskam suggests that this speed is the highest of the design take–off or landing speed of the airplane:

$$V_{tire_{max}} = max \begin{cases} 1.2V_{s_L} \\ 1.1V_{s_{TO}} \end{cases} \tag{5.1}$$

where V_{s_L} is the design landing speed, while $V_{s_{TO}}$ is the design take–off speed for the aircraft.

After the maximum tire operating speed is calculated, tables with tire properties can be consulted (similar to those in [138]) to determine which tire types meet the load and speed requirements.

To determine the Equivalent Single Wheel Loading (ESWL) Roskam suggests the following for tandem twin layouts:

$$ESWL = \frac{P_m}{2} \tag{5.2}$$

where P_m is the load on each main landing gear, defined as:

$$MTOW = P_n + n_s P_m \tag{5.3}$$

where P_n is the load on the nose gear, and n_s is the number of main gear struts (usually 2).

After obtaining the ESWL, the tire choice can be further narrowed down by applying the load/pressure criteria for surface compatibility. And finally, from the remaining tires one is chosen on the basis of

- minimum size
- weight
- wear and tear
- customer preference

Thus the tire diameter, thickness inflation pressure have been determined and it is then possible to determine the Load Classification Number (LCN)[1]from appropriate graphs such as the ones found in Currey[39].

It is now possible to determine the principal dimensions of the tandem twin layout, namely S_T and S_B:

$$S_T = 1.8 \times w_{max}$$

$$S_T = 1.8 \times 1.04 \times w \tag{5.4}$$

where w_{max} is the maximum tire width which is given by Currey to be 1.04 times the tire thickness (w), and

$$S_B = 1.2 \times d_{max}$$

$$S_B = 1.2 \times 1.1 \times d \tag{5.5}$$

where d_{max} is the maximum tire diameter which is given also by Currey to be 1.1 times the tire diameter (d).

[1]Note that the Airport Classification Number (ACN) is a new method that effectively replaces both the LCN and LCG. However, the LCN is still used by many.

The shock absorber stroke is required to calculate the length of the cylinder of the main gear unit. It can be calculated from [138]:

$$S = \frac{1}{\eta_s}\left(\frac{w^2}{1.84g\lambda} - \eta_s S_T\right) \tag{5.6}$$

where:

η_s efficiency factor, the ratio of the Energy absorbed by the tire or the absorber divided by the product of the maximum static load per leg and the maximum deflection or stroke respectively

w the ultimate velocity of descent, specified in airworthiness regulations: FAR 23.473 and 25.473, BCAR Sect. D, Chap. D3-5, Par. 4, and Sect. K, Chap. K3-5, Par. 2. For transport aircraft, $w = 12ft/sec$ $(3.66m/s)$ although in BCAR w depends on stall speed

λ constant, typically $2 - 2.5$ for a transport aircraft [138]

S_t maximum tire deflection

Note that Equation 5.6 represents the projection of the stroke normal to the ground. To the result from this equation, it is customary to add an extra one inch to account for inaccuracies and miscellaneous factors.

The maximum tire deflection (S_t) can be calculated from:

$$S_t = \frac{c\lambda L_w}{p\sqrt{dw_{max}}} \tag{5.7}$$

where:

c constant, equal to 0.5

L_w static load per wheel

p inflation pressure

d wheel diameter, and

w_{max} maximum tire width

Having obtained the shock absorber stroke it is now possible to estimate the length of the cylinder of the main gear unit [39]:

$$l \cong 3 \times S \tag{5.8}$$

The diameter D of cylinder of the main gear unit can be approximated as follows [39]:

$$D = \left(0.5 + 0.3\sqrt{P_s}\right) in \tag{5.9}$$

$$D = \left(1.3 + 0.11\sqrt{P_s}\right) cm \tag{5.10}$$

where P_s is the static load per leg. Equation 5.9 produces the result in inches, and P_s is in lbs, while Equation 5.10 produces the result in centimeters, and P_s is in kg.

Hence all the landing gear characteristics shown in Figure 5.2 can be calculated.

5.4 Interaction with the Aerodynamics Module

The aerodynamic design module and the landing gear preliminary design module are strongly coupled. The landing gear module provides output that enters directly into the aerodynamic module optimization loop (See Appendix F). By treating the landing gear module output as a constraint, the fillet surface is required to produce the necessary volume for the gear retraction and only then can it continue the optimization loop. There are no feedbacks from the aerodynamics module into the landing gear module, since the creation of an

optimization process within the module is considered beyond the scope of this thesis. For a more detailed and thus more realistic situation, such a feedback would account for the links between interference drag and overall drag (by linking the size of the fuselage to the landing gear requirements). The overall drag, in turn, would influence the take-off mass of the aircraft, and would thus affect the entire preliminary analysis.

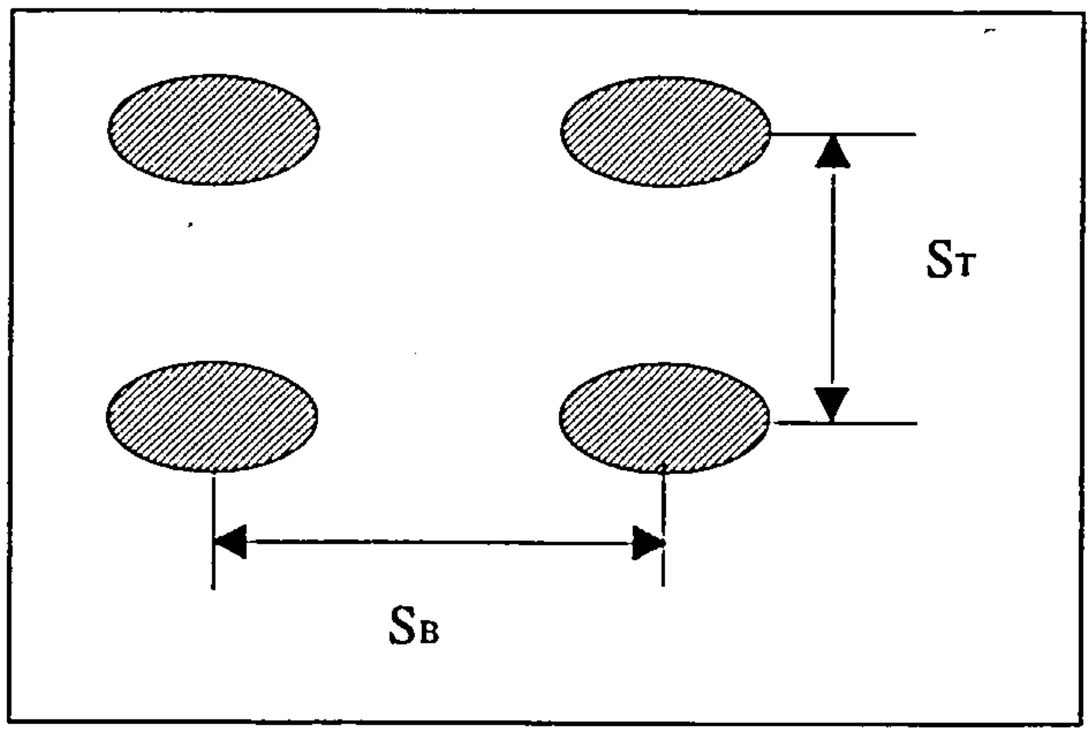

Figure 5.1: Dual Tandem Landing Gear Wheel Layout

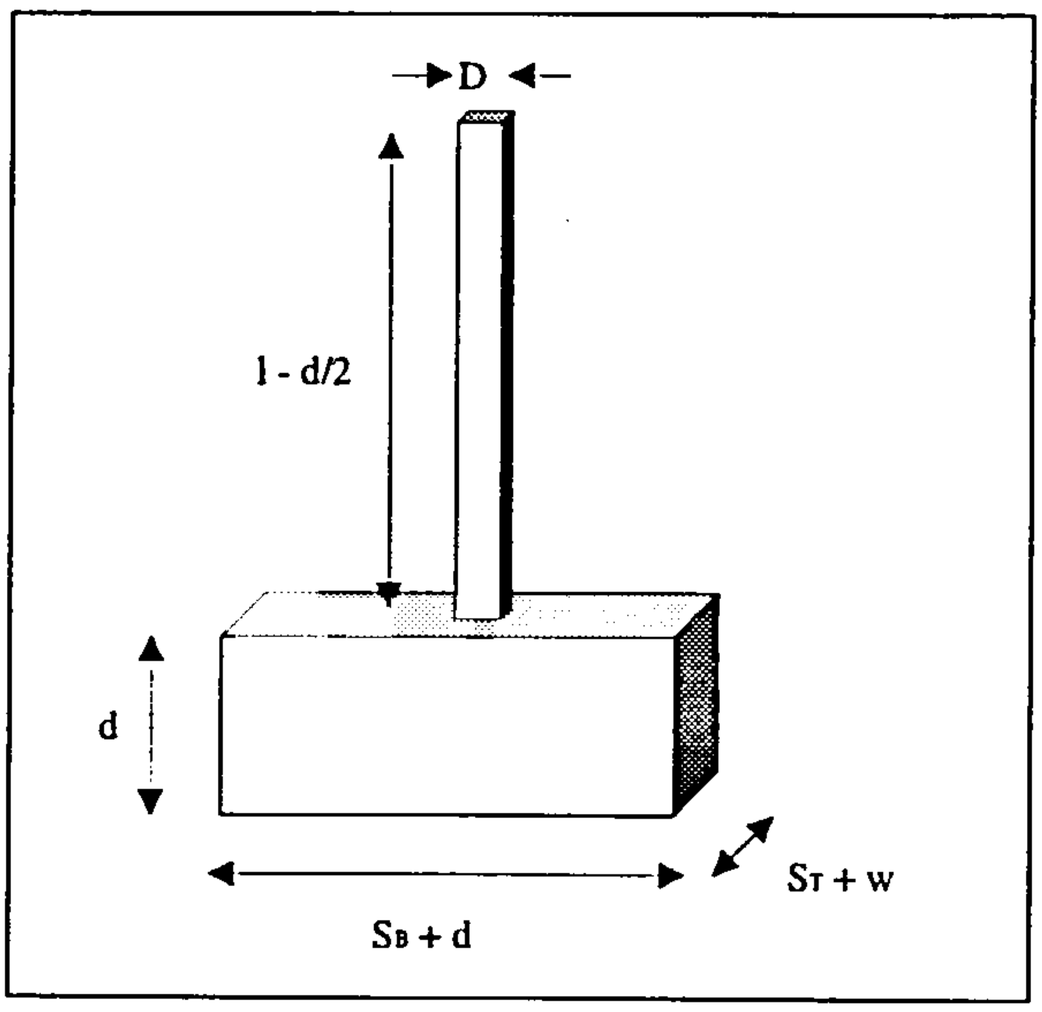

Figure 5.2: Calculated Main Landing Gear Features

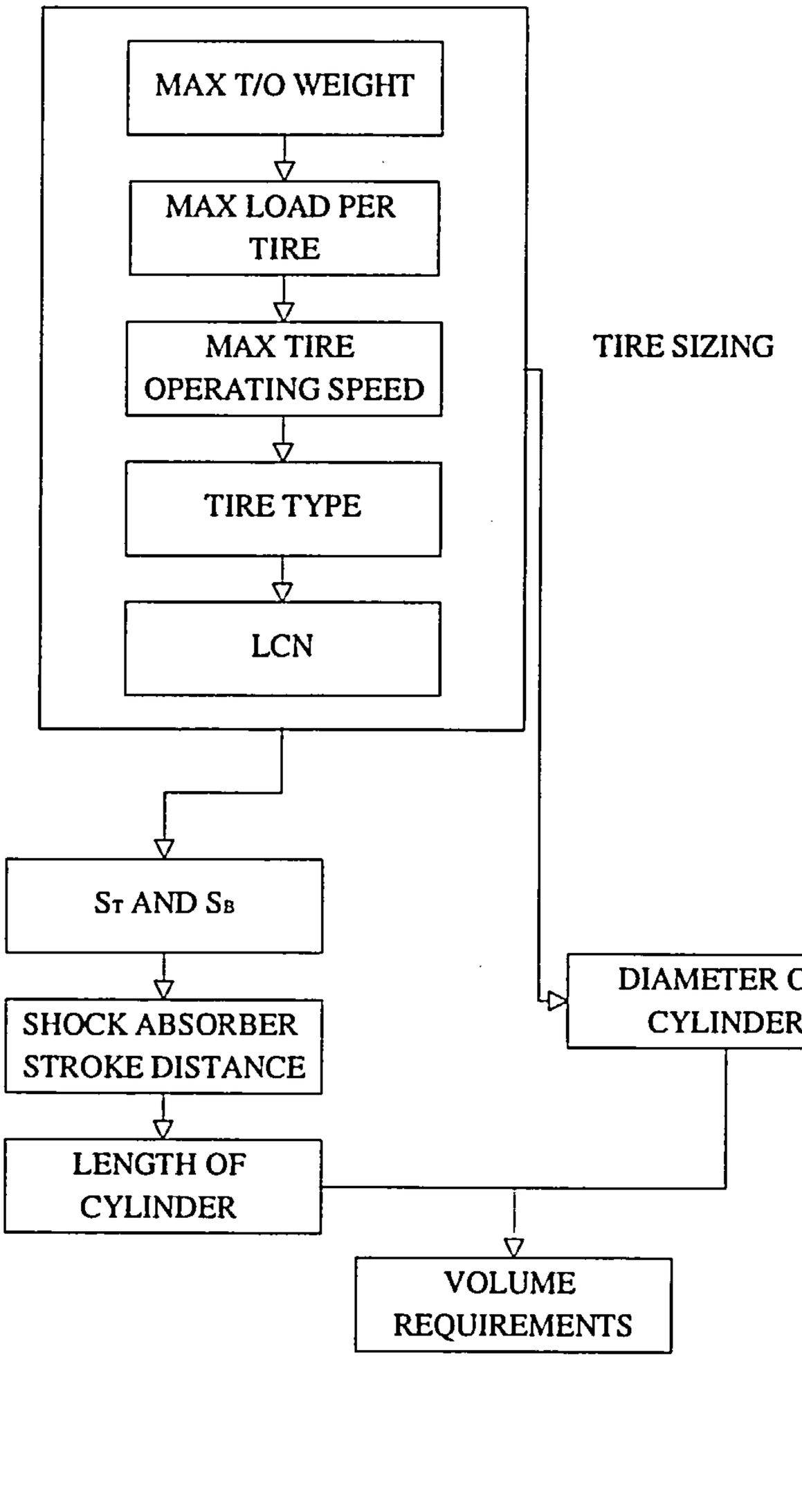

Figure 5.3: Landing Gear Module Block Diagram

Chapter 6

Integration and Optimization

6.1 Introduction

Attempting to determine the minimum value of the interference drag over a series of configurations with the minimum number of point simulations is not a trivial task. It is important to determine from the outset the conditions over which the result of such an effort would be acceptable. It is also important to avoid the pitfalls that may affect the outcome of the optimization. With this in mind, the first thing that must be defined before any such attempt is made, is the objective function (or response function).

6.2 Defining the Objective Function

The interference drag can be derived from the results of the EAGLE TLNS solver (See Appendix 4). The force is normalized by the interference drag obtained by the EAGLE TLNS run for the aircraft with no fillet present. Therefore, at this stage, the objective function is defined as the ratio of the interference drag of a given configuration to the interference drag forces of the aircraft with no filleting.

After all the required values have been obtained by the Design of Experiments method in use, it is common practice to subtract the mean of these values from each one. The normalized values are then used to produce the quadratic response surface. If extra point evaluations are available they could be used to estimate the variance that results from the response surface fit as a direct result of the lack of fit. If, on the other hand, one uses an optimized design to produce the values, depending on the design, it could be possible for the response surface fit to exhibit the least variance. Since the hybrid design is near-rotatable and was constructed to satisfy the $X'X$ criterion, it is expected to exhibit such a behaviour.

The quadratic response surface from this point on becomes the objective function. If the stationary point lies within the feasible domain, then this is the desired optimum point. If on the other hand, it lies outside the feasible region, the use of an optimizing method is required.

6.2.1 Input to the Optimizer

The required inputs for the complete optimization to take place can be classified into Aerodynamic Input and Interdisciplinary Input, the latter represented in this thesis by the Landing Gear Input.

6.2.1.1 Aerodynamic Input

The EAGLE TLNS code requires a number of characteristics of the flow region. These characteristics must be decided upon before any simulation is performed, and must remain constant until the end of the simulation, unless any of them are input (design) variables. Such characteristics of the flow region are the Mach number, the CFL number, the freestream designation on computational block surfaces.

In addition, in the design variables for the example six variable optimization study presented in this thesis, a number of Aerodynamic factors must be determined from the conceptual design of the aircraft that will affect the values of these design variables. Such sensitive variables are the span of the wing, the root chord, the spanwise chord distribution, the airfoil section required.

In the six variable demonstration, the fillet surface is represented by a pair of Bezier point nets (See Appendix C). The nets are nonuniform, and consist of 5×4 control points. The fillet-wing intersection line for the upper wing is represented by five control points. These five points fully define a space Bezier curve. The same is true about the fillet-fuselage intersection line. The surface is derived by the Tensor Product method from these two curves. The lower fillet surface is formed following the same method.

6.2.1.2 Landing Gear Input

The volume the fillet surfaces enclose is sent to the Landing Gear module as soon as the surfaces are defined. The module determines if the appropriate volume required for the landing gear retraction is available. If the volume is not available, the lower fillet surface is modified, redefined, and re-evaluated by the volume requirement routine of the Landing Gear module (See program flow in Appendix F). If the volume requirements are satisfied, the fillet surfaces are incorporated into the surface geometry definition file, and the grid generation module uses it to produce the grid.

The inputs for the Landing Gear module are fixed (See Chapter 5). No information is required to be transferred back to the Landing Gear module after the decision on the volume requirements is made. A more realistic module may require some feedback (i.e. the extend along the span over which the lower fillet will be defined), or may even contribute design variables for the optimization.

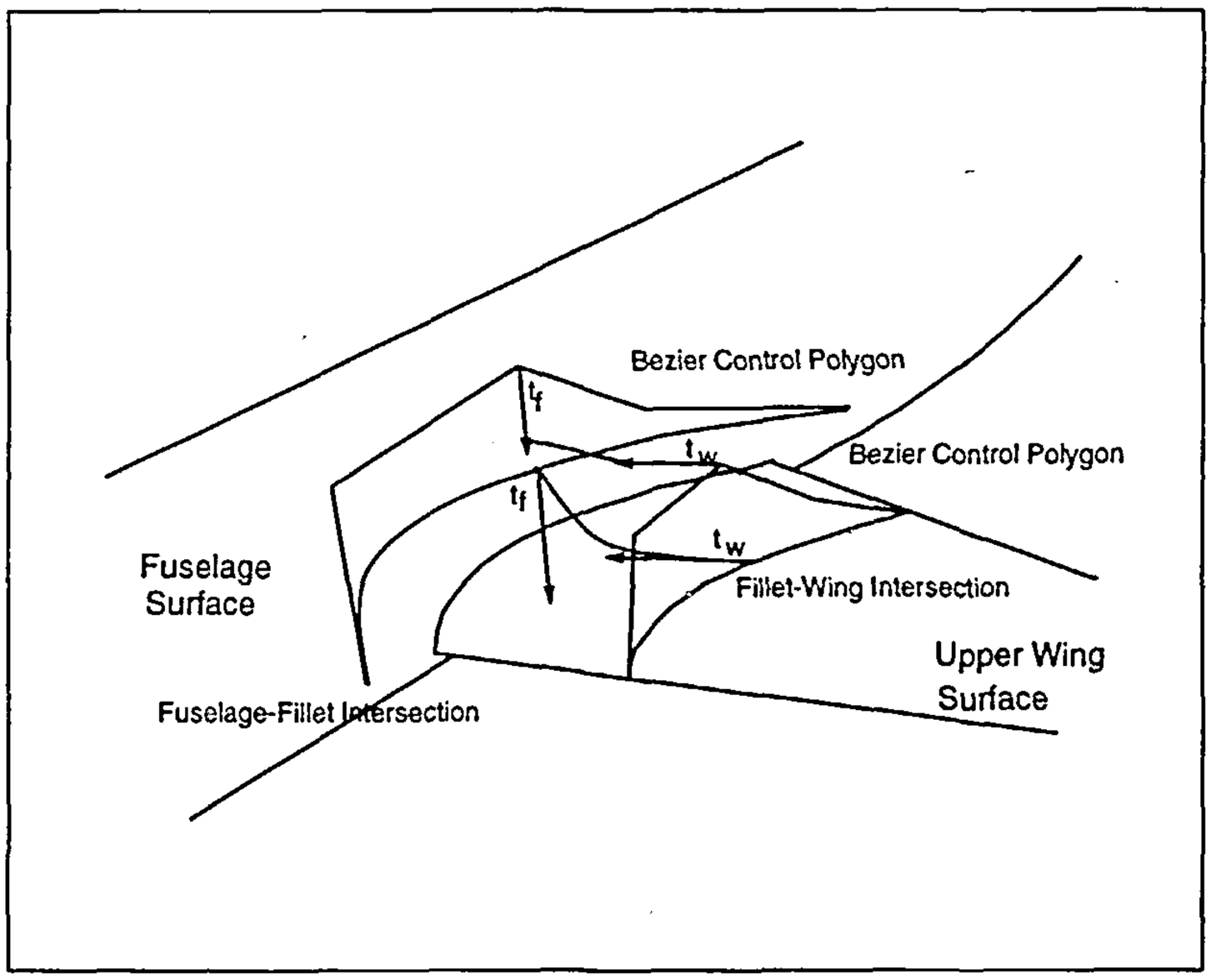

Figure 6.1: Tangent Directions for the Six Variable Example

6.3 Design Case Description

In the general case, the 5×4 control net of Bezier points contains ten Bezier points that are fixed (the five points defining the fillet-fuselage intersection line, and the five defining the fillet-wing intersection line) and ten points with coordinates that can vary. In the particular test case used in the presentation of this thesis, these ten points are constrained as follows.

In Figure 6.1, t_w is the local tangent at the fillet-wing intersection line at the point located at the same chordwise coordinate as the third control point of the Bezier polygon, in the spanwise direction, towards the fuselage. Also, t_f is the local tangent at the fillet-fuselage intersection line at the point located at the same position along the length of the fuselage with the third control point of its own Bezier polygon, in the circumferential direction, towards the wing-root section. To ensure a smooth transition from the fuselage to the fillet and the

fillet to the wing, these tangents are also enforced on the corresponding control points of the Bezier polygons. Since the surface is defined by a nonuniform 5×4 control net, by repeating the same procedure appropriately on all the control points of the two bounding curves (the fillet-wing intersection and the fillet-fuselage intersection) the internal ten control points are constrained to move each one along a line in space parallel to the local tangents at the corresponding intersection curve points.

It is then possible to control the location of these points on the tangent lines with only a single parameter. The tangency assumption, therefore, reduced the number of variables to control from 3×10 (three coordinates to be determined for each internal control net point) to 10 (one parameter per internal control point). To further reduce the variables, the following assumptions are made.

The three internal control points towards the fuselage C.3.2 will all have a unique parameter value. The three internal control points towards the wing will also have a unique parameter value. Finally, the four outer points, controlling the leading and trailing edge shapes will have their own unique value.

The parameter values were defined for the first three control points, in terms of the vertical distance between the relevant fillet-fuselage control polygon point and the corresponding fillet-wing control polygon point. It was apparent that for values between 10 and 50, the resulting surface exhibited a very smooth behaviour between the bounding curves and control polygons, while between 2 and 10 the fuselage tangency condition would not become apparent and relatively sudden changes in slope would occur. These values remained within the range of the parameter out of curiosity about the behaviour of the flow around them. Values of the parameter greater than 50 would result in a very abrupt change of direction on the fillet surface (i.e. from the horizontal attitude leaving the fillet-wing intersection, to adopt to the vertical attitude, so that it will blend with the fillet-fuselage intersection).

The parameter values for the second three control points were defined in terms of the horizontal distance between the relevant fillet-wing control polygon point and the corresponding fillet-fuselage control polygon point. In this case, values between 2 and 50 produced very smooth surfaces. Again, values greater than 50 would result in a very pronounced "corner" in the surface.

Finally, the parameter values for the third set of points ranged from 2 to 200 producing a very smooth, almost straight line between the trailing edge of the fillet-fuselage intersection curve and the fillet-wing intersection curve, to a very pronounced "corner" similar to the previous discussion, respectively.

Remaining parameters that entered in the optimization were the spanwise distance of the fillet-wing intersection, the scaling factor applied to the wing-root section in the z-direction to produce the circumferential location of the fillet-fuselage intersection line, and the scaling factor applied to the chord of the wing-root section along the length of the fuselage, to produce the location of the fillet-root leading edge and trailing edge. Limiting values for these variables were considered three times the half-thickness and twice the chord with minimum values the unity, i.e. no scaling.

6.4 Classical Optimization Components

It is possible after the response surface fit, the stationary point not to lie within the feasible region, as was the case in the six variable optimization example described in the previous section (See Appendix F for specific values). It is necessary then to determine the minimum area of the response surface within its feasible region subject to constraints.

6.4.1 Classical Optimization Methods

Since the response surface is a second order surface, to determine its extremes within a given feasible region is possible using a classical gradient optimizer. If, furthermore, the optimizer accepts second derivative information, the code will avoid saddle points and reach the minimum region of the surface. Repeating the optimization with other starting points will demonstrate whether the local optimum appears to be a global optimum always within the feasible region. In the six variable problem, the global optimum was found consistently with a number of different starting points with combinations that either belonged to the Experimental Table or not.

6.4.2 The NPSOL Optimizer

The classical optimizer used to determine the region of optimality in the response surface when the stationary point was shown to lie beyond the response surface's feasible region was NPSOL (See Appendix E, a program designed to minimize smooth functions subject to a number of linear or nonlinear constraints. It uses a quadratic programming algorithm, and implements first and second derivative information.

6.4.3 Closing the Loop

The final decision must be if the result of the optimization is satisfactory. Depending on the experimental design used, it may be possible that the second order surface does not exhibit a good fit to the experimental data. One method to check for the goodness of fit is to obtain more experimental points and calculate the variance due to bad surface fit [50]. The decision must be made then if the approximation will be acceptable or not. In the six variable design, the minimum of the response surface as shown at the end of Appendix

F is only 2 percent different than the estimated value given by the response surface. Therefore, the result is considered acceptable. If, however, after the final EAGLE TLNS run, the discrepancy between the results is too large to dismiss as approximation error, two options are available. The first is to choose a different experimental design satisfying an optimality criterion that minimizes the variance of the second order surface fit and repeat the surface design process from the beginning, having this time extra information to evaluate the goodness of the final response surface fit from the previous run, or perform a second optimization process around the area of the response surface minimum to determine more accurately the behaviour of the second order surface. In the latter case, only a subset of the feasible region is examined. By locating the minimum of this first order surface and determining the goodness of fit of this surface it may be possible to improve the accuracy of the method. If the discrepancy between the predicted response of the objective function and the actual value of the function at the predicted optimum is still unacceptable, a subsequent first order optimization may have to be performed.

As in any constrained multivariable optimization process, there is no guarantee that the global minimum will be reached. However, previous knowledge of the objective function behaviour is essential: if the function is not smooth over the region of interest, the response surface will not produce reliable estimates. The region must be small enough that the objective function can be assumed as smooth over it. After a satisfactory response surface fit is obtained, if the stationary point lies outside the region of interest, this region of interest can be expanded, and based on the optimizer information, a new region of interest can be chosen relying on the minimization information provided by the optimizer.

6.5 Optimization Results

A dramatic reduction in total interference drag in the junction was observed after the conclusion and evaluation of the optimization run. The agreement between the expected values of the incremental drag obtained by the response surface model and the actual drag numerically calculated by the EAGLE TLNS flow solver was more than ninety percent. This result agrees with the assumptions of regional near-linearity in the objective function behaviour for simple geometric definitions for the fillet surfaces used by a number of industries (See Chapter 3).

By far the most interesting results arise when the coefficients of the interaction terms in the response surface formulation are studied. From Appendix F, the interaction term coefficients are:

$$\begin{bmatrix} 0.0020289 & -0.00091602 & 0.0 & -0.021468 & -0.0096275 & 0.0062781 \\ -0.000916 & 0.0020402 & 0.0 & -0.021757 & 0.0097442 & 0.0066748 \\ 0.0 & 0.0 & 0.000161 & 0.0017259 & -0.0018593 & 0.00012172 \\ -0.021468 & -0.021757 & 0.0017259 & 6.9372 & 2.8286 & -0.33005 \\ 0.0096275 & 0.0097442 & -0.001859 & 2.8286 & 1.2806 & 0.22908 \\ 0.0062781 & 0.0066748 & 0.0001217 & -0.33005 & 0.22908 & 0.10499 \end{bmatrix}$$

It can immediately be observed that there is strong interaction between the fourth and fifth column and the third and fourth row of the array, corresponding to the chordwise and thickness-wise scaling of the root fillet section. It can also be observed that there is essentially no interaction between the first and second column of the third row and the first and second row of the third column, implying that very weak interactions occur between the two fillet surface shape factors. Furthermore, judging by the remaining terms in the third row, one would expect additionally a very weak interaction of the third variable, the fillet leading and trailing edge shape parameter.

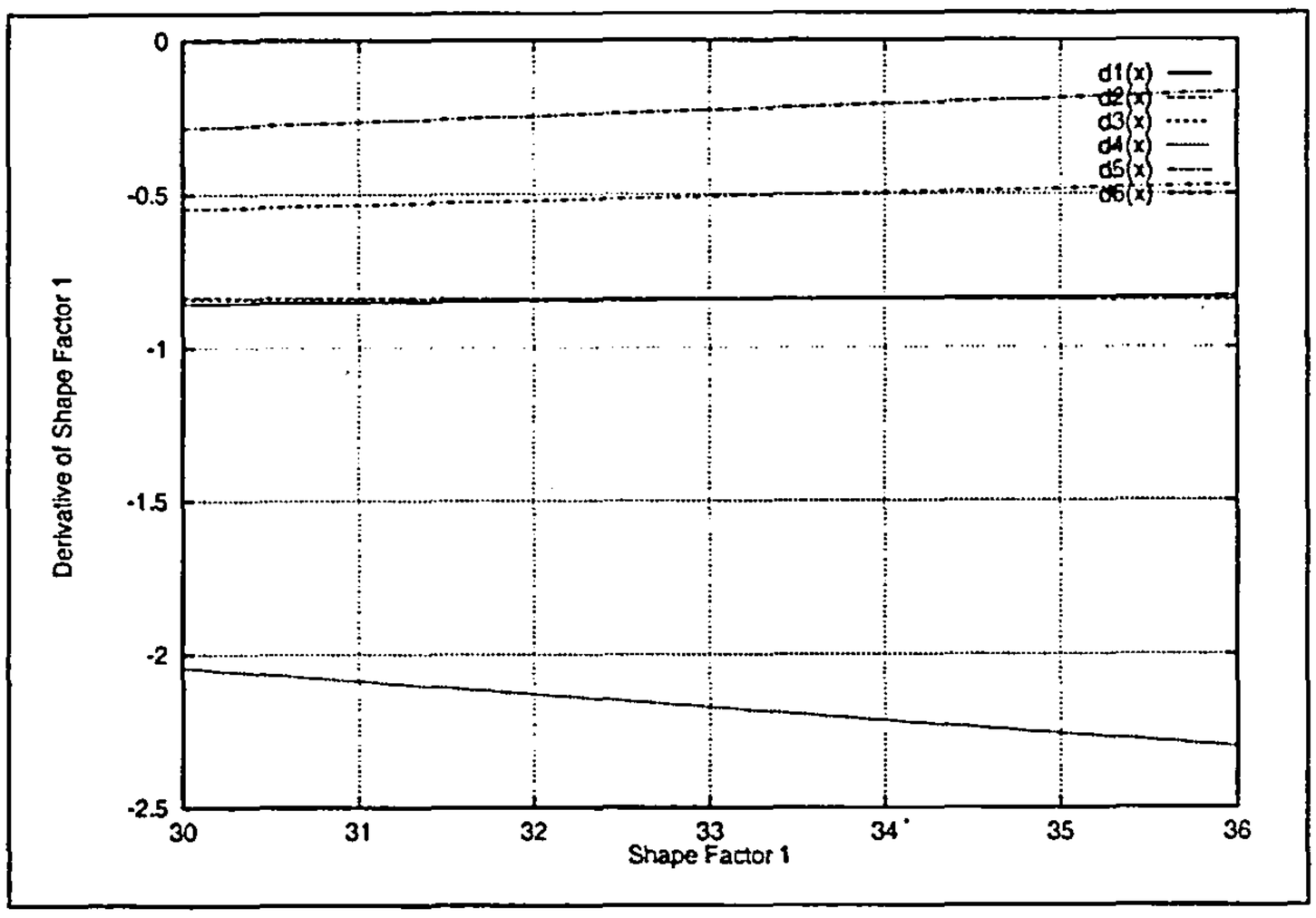

Figure 6.2: Sensitivities for Fillet-Fuselage Surface Shape Factor

To verify the above observations, a sensitivity analysis was performed in the region of the optimum. The gradient of the response surface formulation was obtained, and resolved into its components close to the optimum region. For Example, Figure 6.2 indicates the behaviour of the derivative of the objective function (the response surface formulation) with respect to the first variable, i.e. the fillet-fuselage junction shape factor. When all the variables are perturbed, one at a time, the lines shown in Figure 6.2 can be obtained. The magnitude of the y-axis denotes the values of the partial and how it will change when each one of the variables will be perturbed around the optimum. The same analysis was performed for the partial derivatives of the objective function with respect to all six variables and the results are shown in Figures 6.2-6.7.

From the sensitivities graphs, the conclusions obtained from the observation of the interaction term factors were verified. In all the graphs, the first and second variables, namely the fillet-fuselage junction shape factor and the

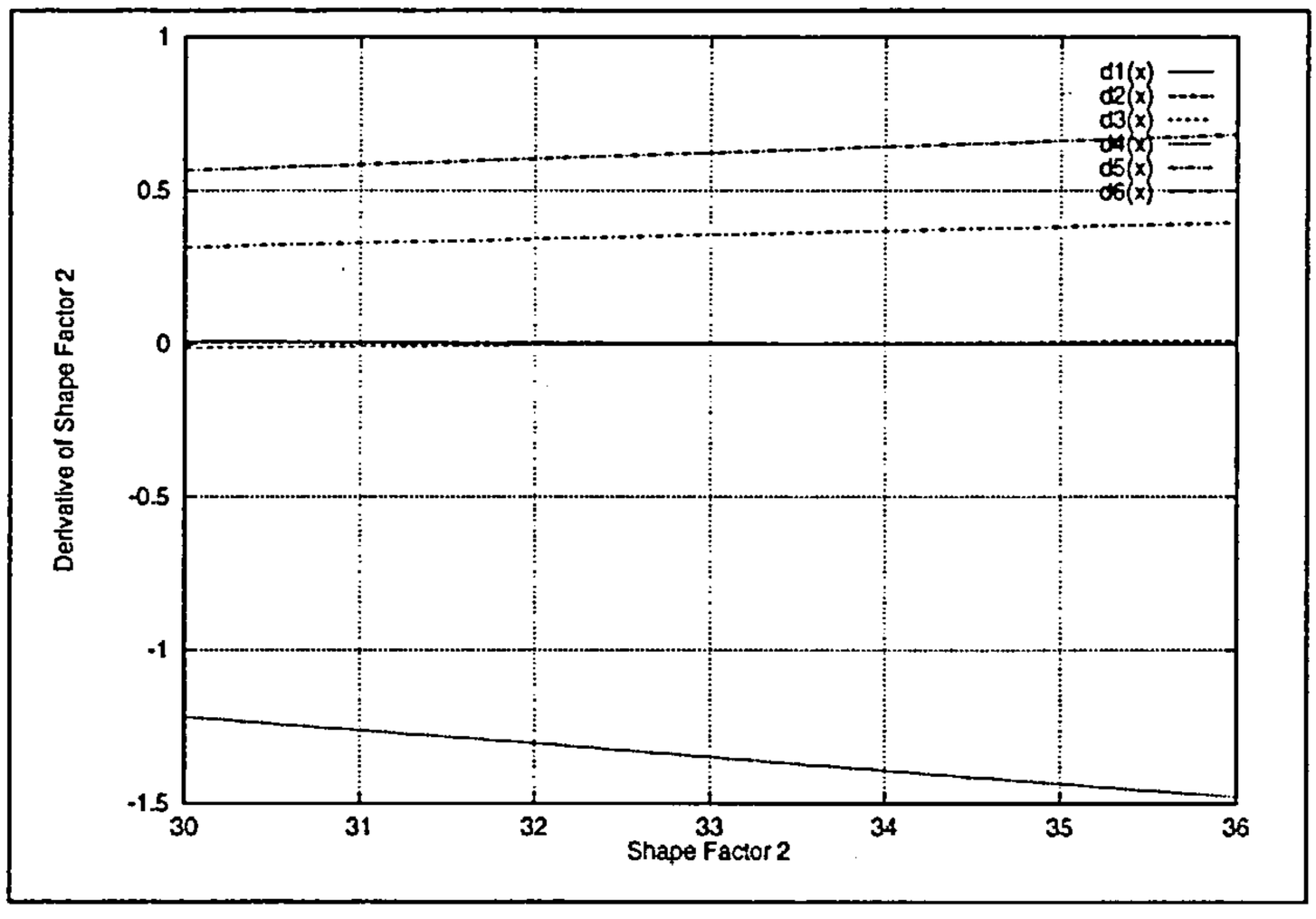

Figure 6.3: Sensitivities for Fillet-Wing Surface Shape Factor

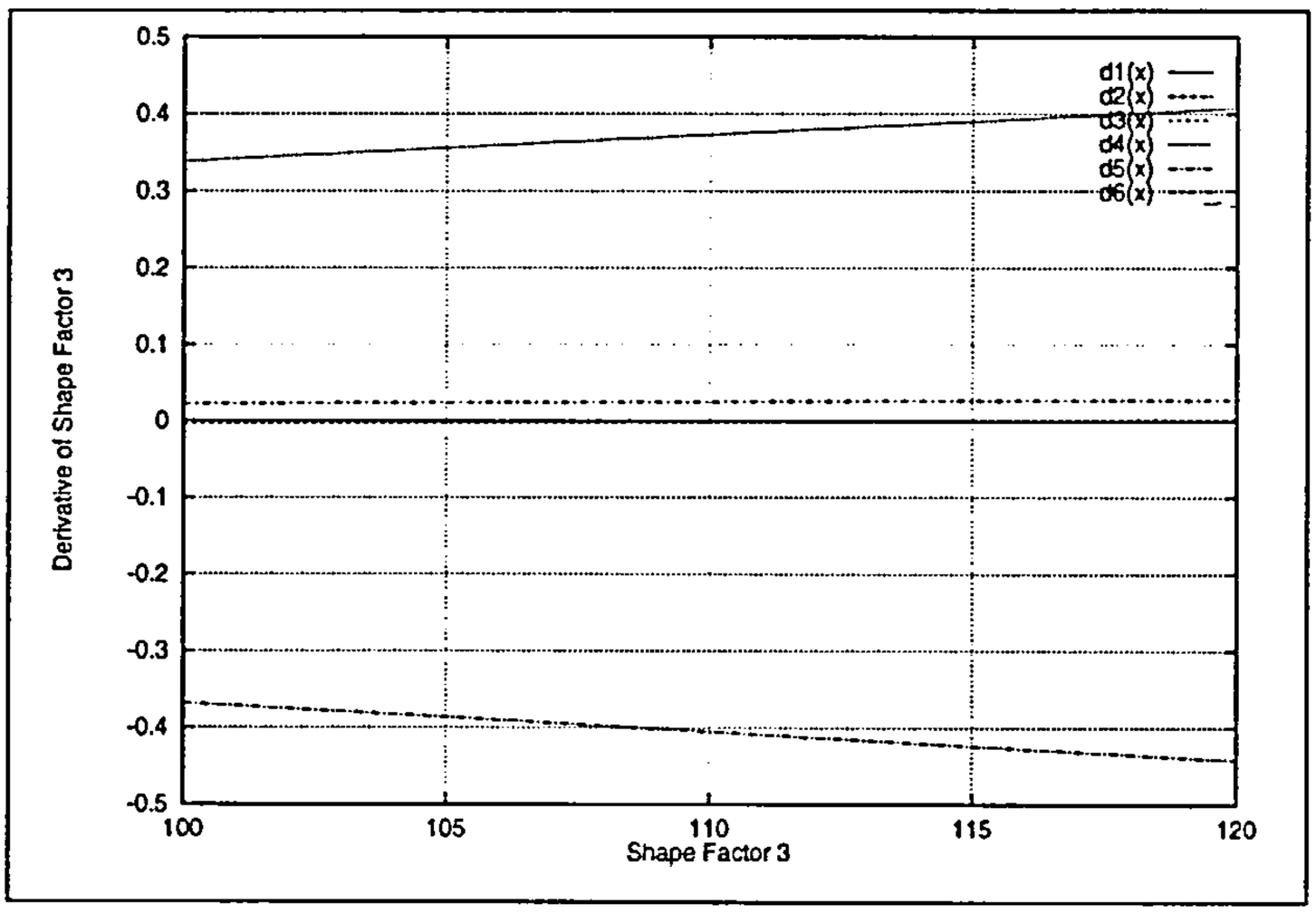

Figure 6.4: Sensitivities for Fillet Leading and Trailing Edge Shape Factor

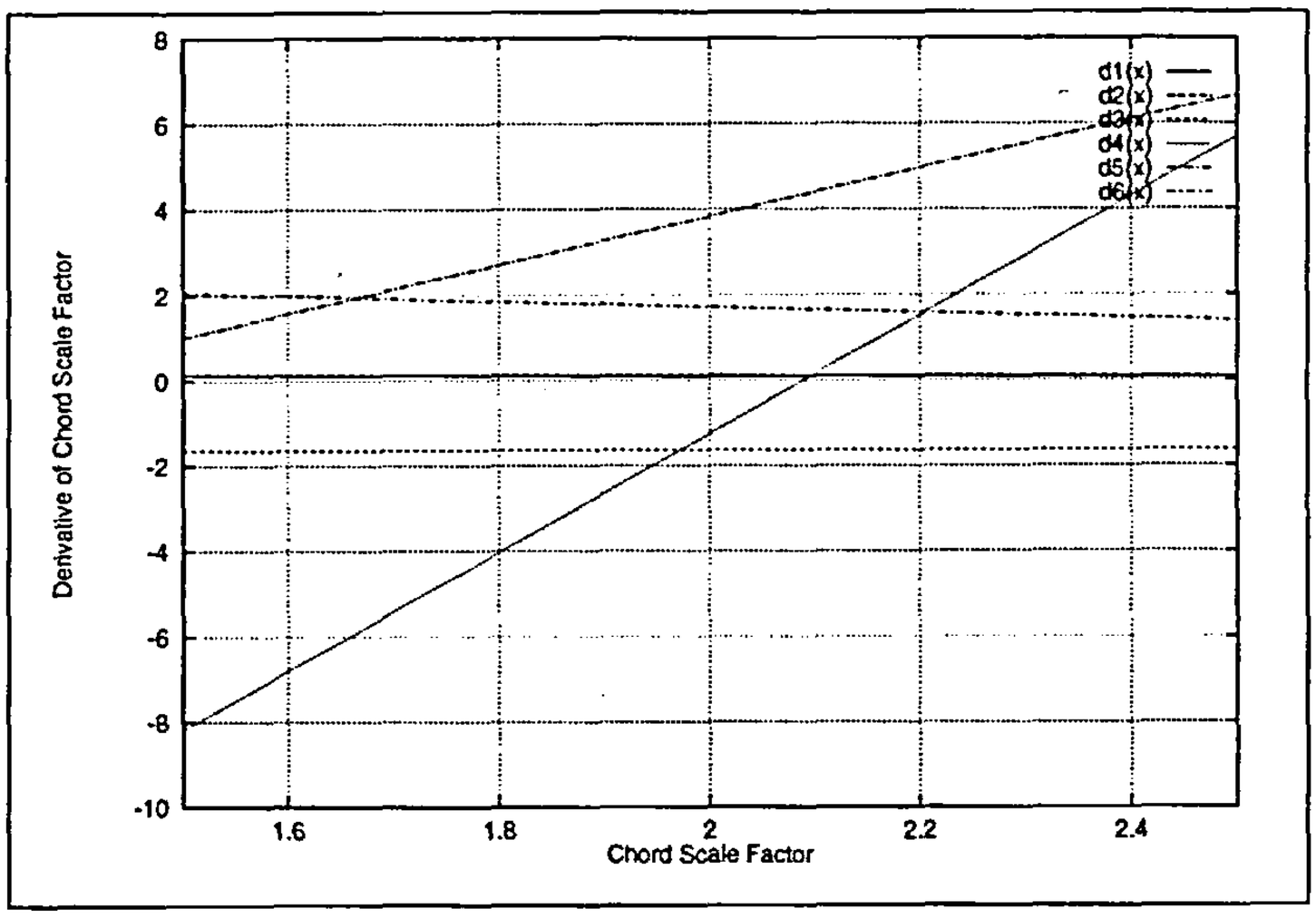

Figure 6.5: Sensitivities for Fillet Root-Section Chord Scaling Factor

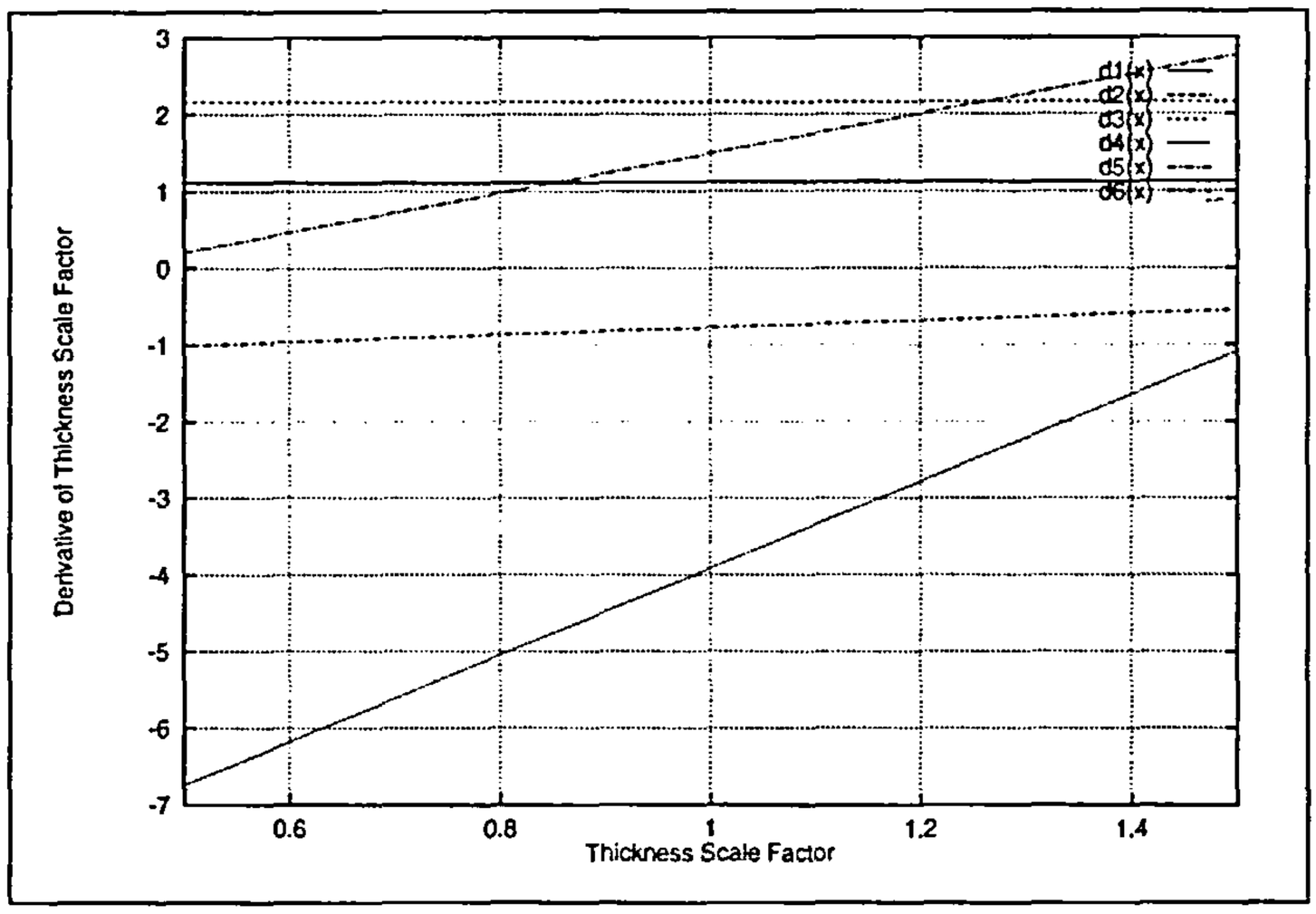

Figure 6.6: Sensitivities for Fillet Root-Section Thickness Scaling Factor

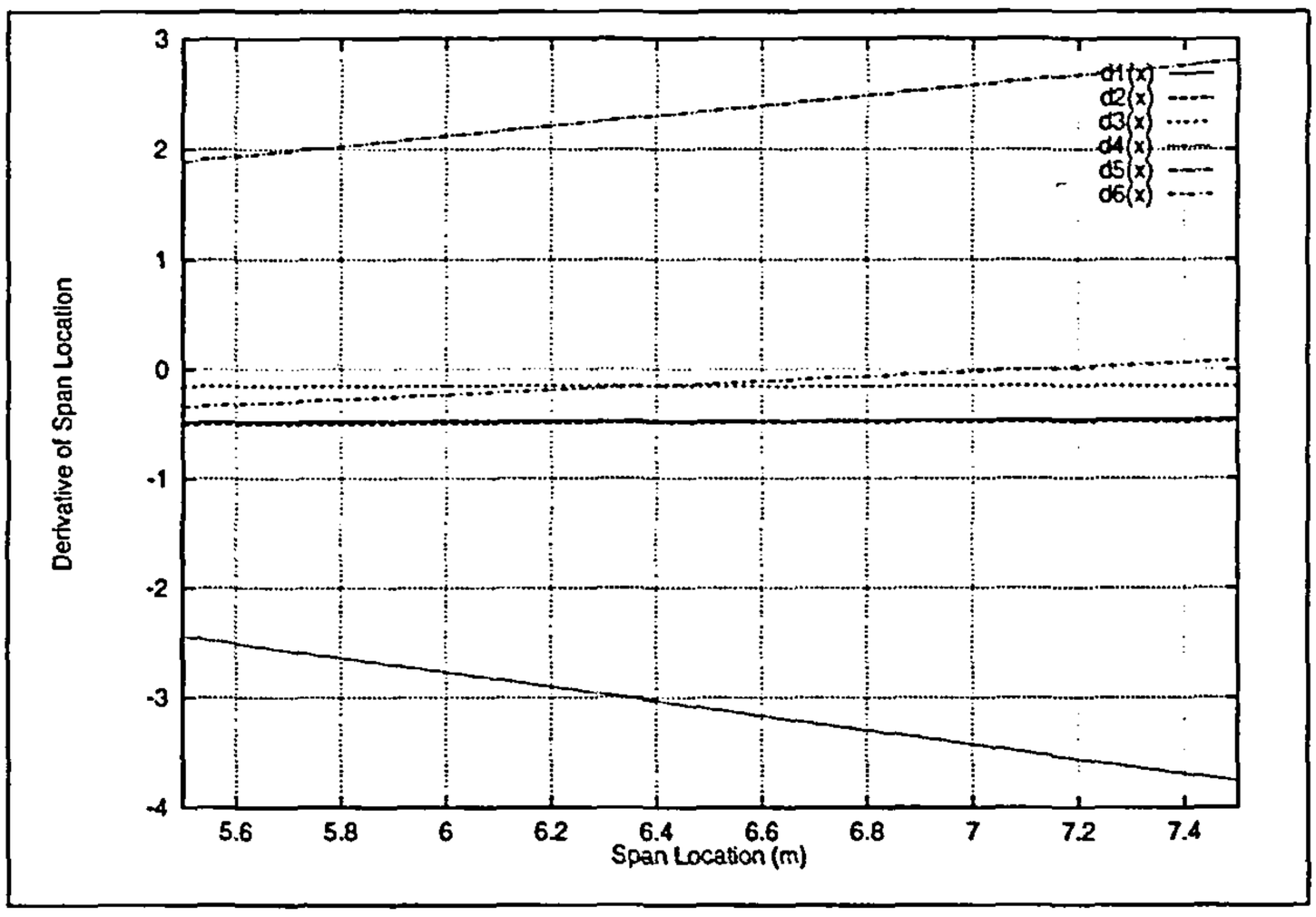

Figure 6.7: Sensitivities for Fillet-Wing Junction Location

fillet-wing junction shape factor ($d1$ and $d2$) do not influence significantly the gradients of the objective function. Also, from Figure 6.4, it is evident that the leading and trailing edge fillet shape factor causes the smallest changes in the gradient components, with a very small sensitivity to only the chord scaling factor and to the thickness scaling factor for the fillet root-section.

On the other hand, very strong gradients are generated by perturbing the chord scaling factor for the fillet root-section. It produces the highest derivatives in all graphs, and particularly in Figure 6.5 it is evident that the gradient with respect to the chord scale factor will produce large derivative values that will be increasing for increasing magnitude of perturbations towards the feasible region. Not as large perturbations will be created for the same gradient by changes in the thickness scaling factor. Nevertheless, the value of the gradient will either increase or decrease, depending on the direction of the change. No other variable changes influence so profoundly the behaviour of this gradient.The fact that the chord scale factor and the thickness scale factor are

dominant agrees with the guidelines given by Hoerner [77].

The remaining variable, i.e. the spanwise location of the fillet-wing intersection does not produce large gradients for changes of the variables around the optimum. Once more, the largest changes in this gradient are produced by changes in the chord and thickness scale factors, with insignificant changes due to perturbation of the remaining variables.

Chapter 7

Results and Discussion

7.1 Flow Solver Results

In this section, the results of the EAGLE TLNS flow solver are described. The pressure contour plots accompany the rendered images of the pressure distribution on the upper, lower wings and the upper fuselage. Before any discussion on the results, though, the optimized fillet is presented below.

The optimized parameters for the fillet were used for one last simulation of the TLNS code. The TLNS code was given a Mach number of .85 and a Reynolds number of 8.5×10^7.The shape of the upper surface of the fillet is shown in Figure 7.1 and its lower surface is shown in Figure 7.2.

The pressure distributions on selected locations on the upper and lower surface of the wing are shown in Figures 7.3, 7.4, 7.5, 7.6 and 7.7 respectively. These locations are given in terms of wing span fractions, and are intended to show in a clear way the physics of the flow in the junction region. The rendered versions of the pressure contours for the upper wing are shown in Figure 7.8 and for the lower wing in Figure 7.9. Finally, the rendered version of the upper fuselage pressure distribution is shown in Figure 7.10.

Further examples of configurations producing high are provided in Ap-

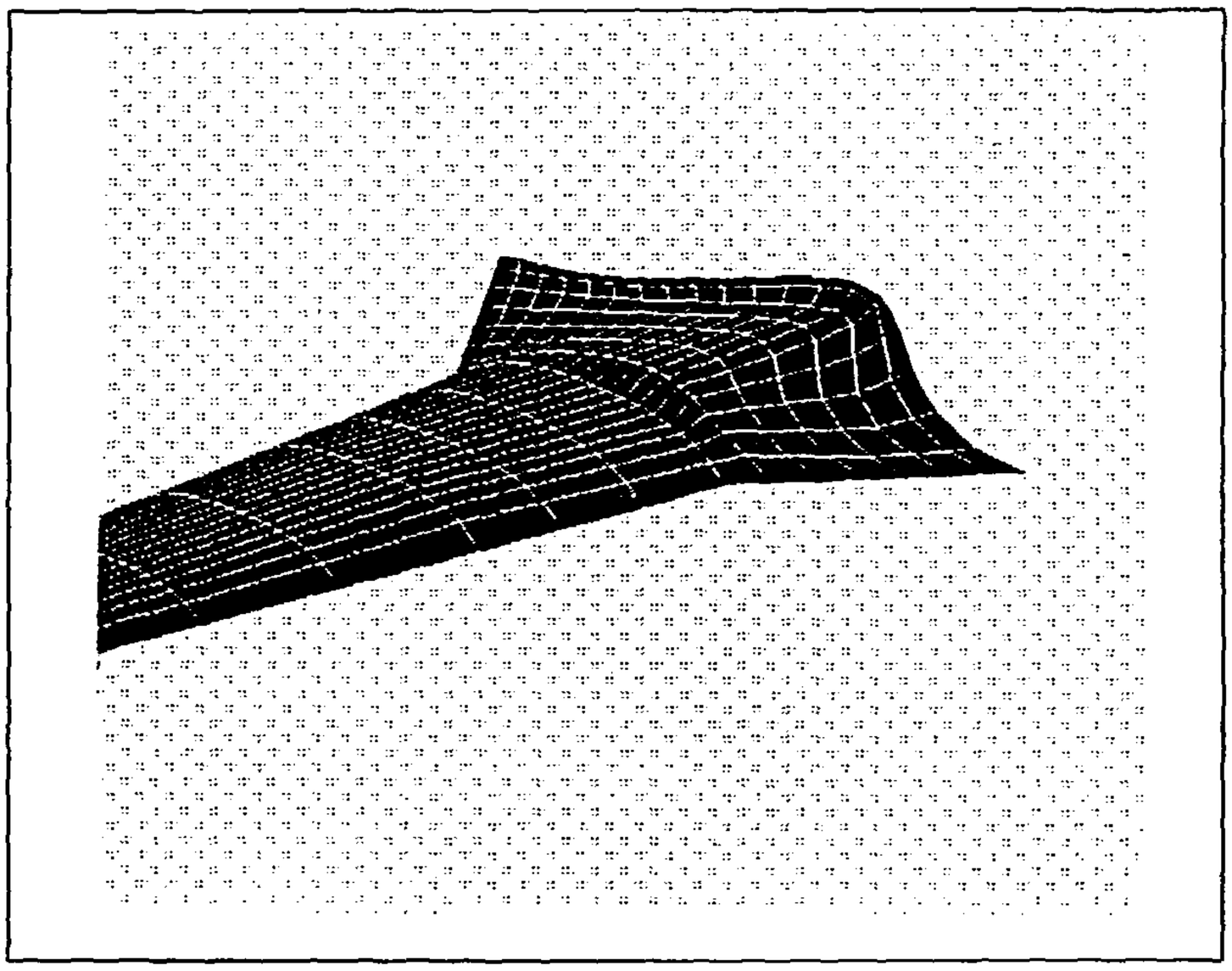

Figure 7.1: Upper Surface of Optimized Fillet

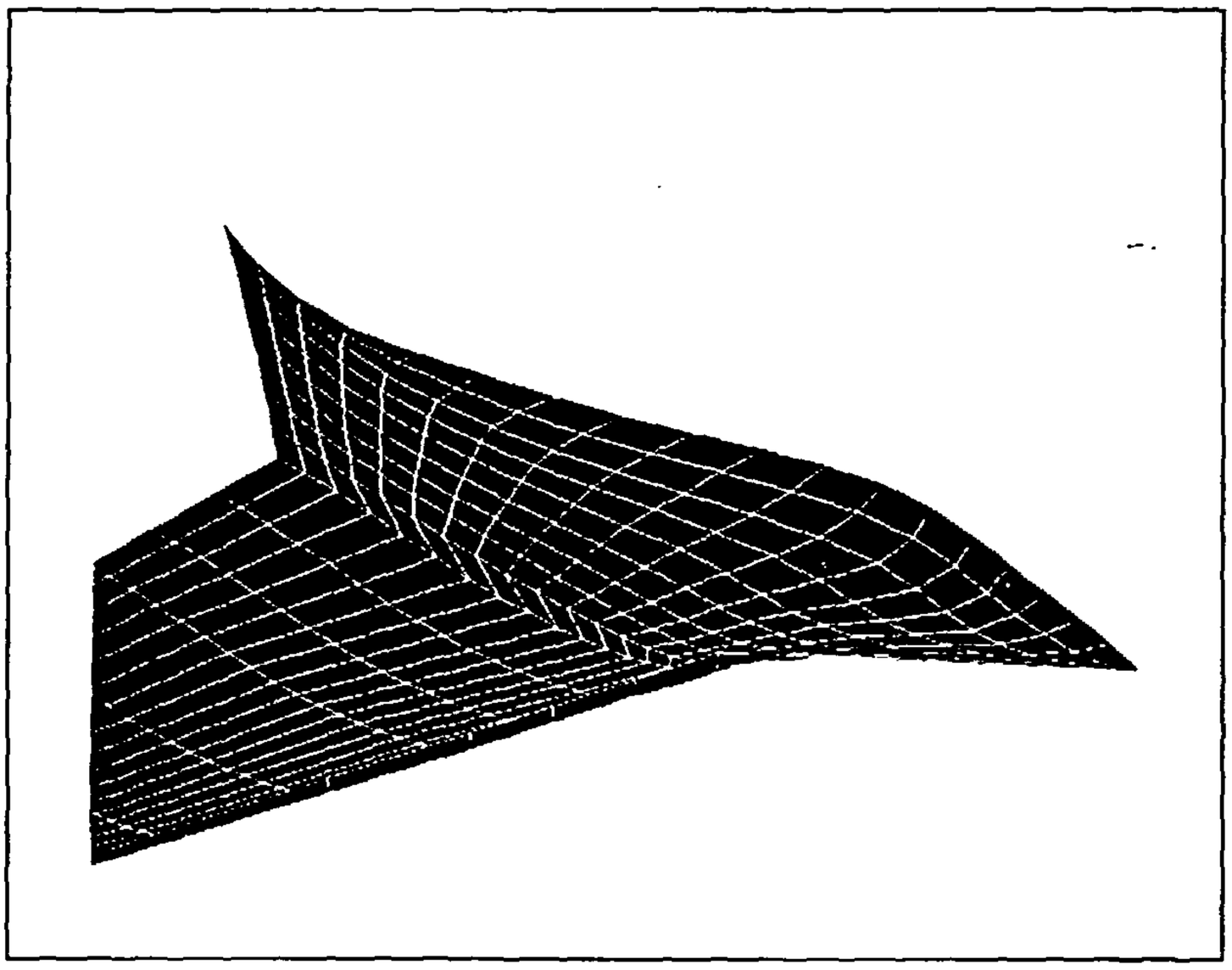

Figure 7.2: Lower Surface of Optimized Fillet

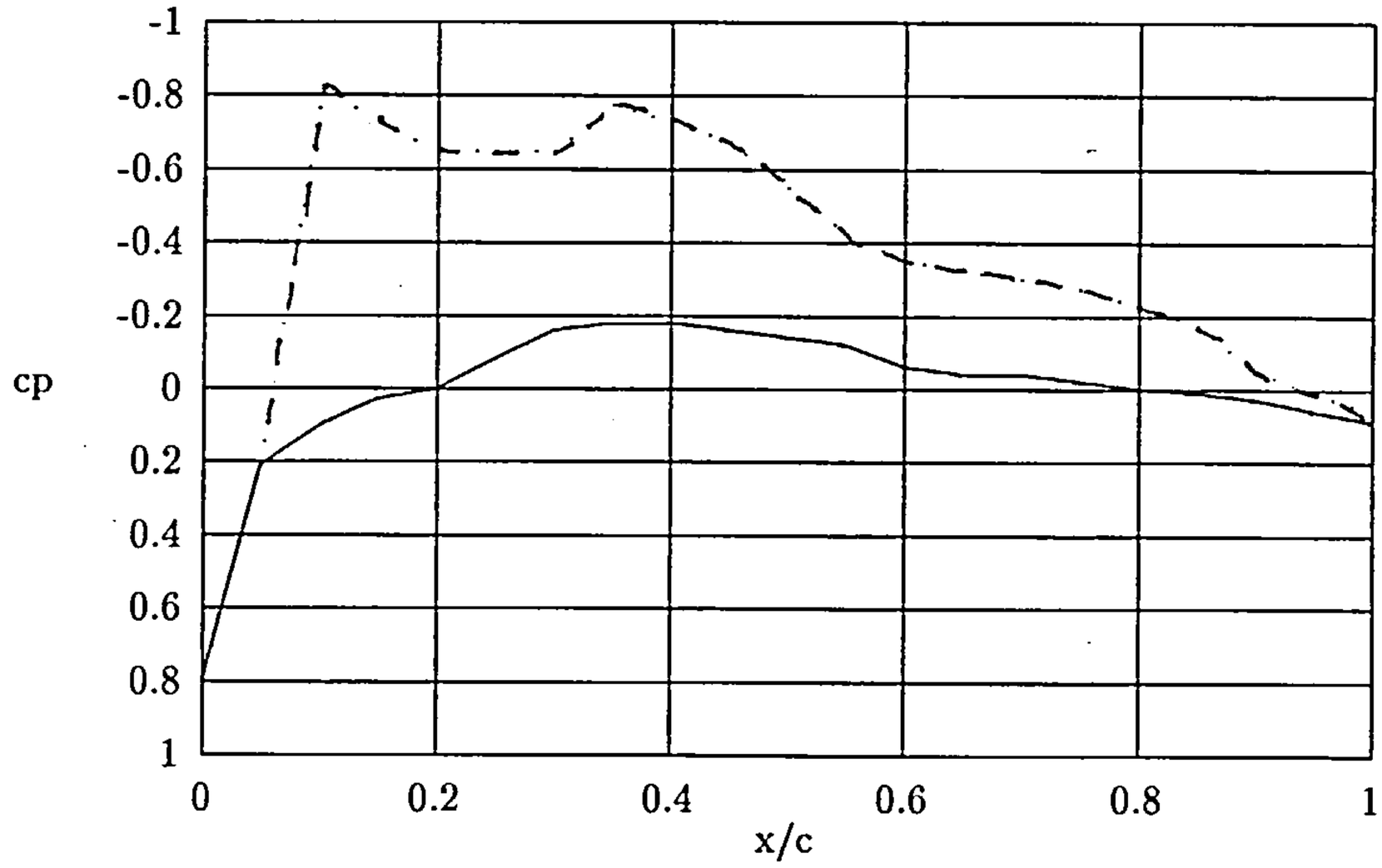

Figure 7.3: c_p distribution at $Y/S = 0$ for optimized fillet

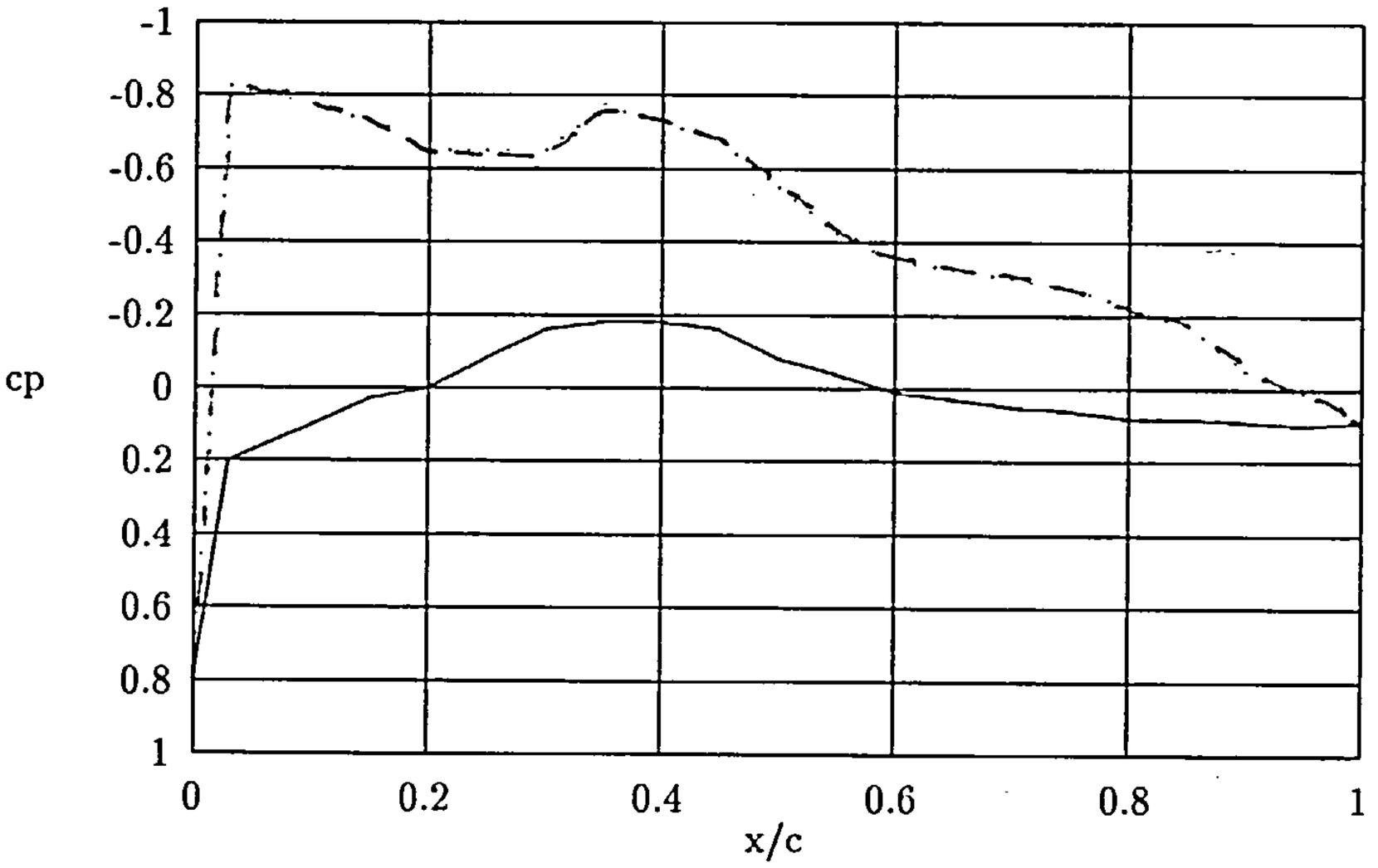

Figure 7.4: c_p distribution at $Y/S = 0.1$ for optimized fillet

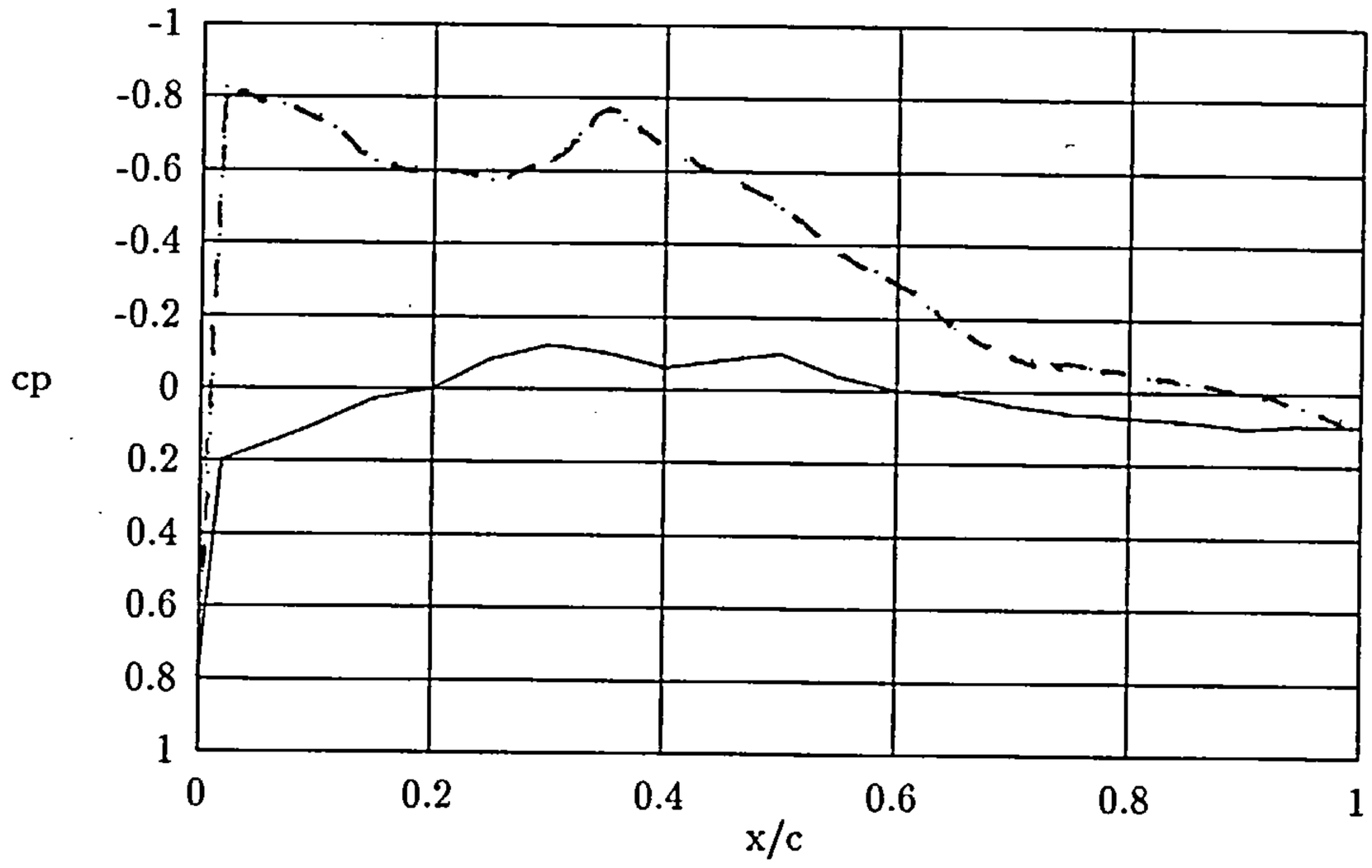

Figure 7.5: c_p distribution at $Y/S = 0.5$ for optimized fillet

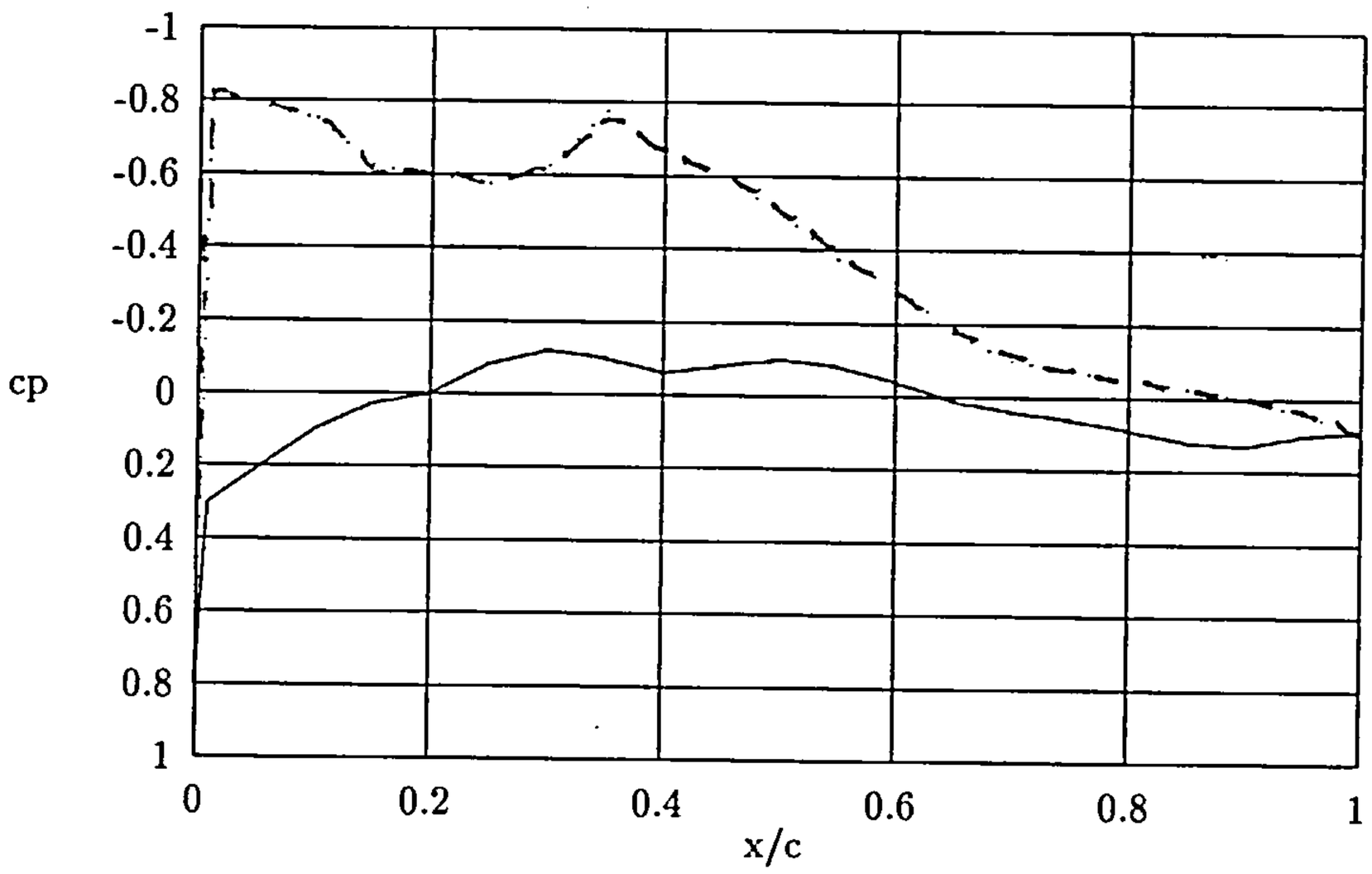

Figure 7.6: c_p distribution at $Y/S = 1$ for optimized fillet

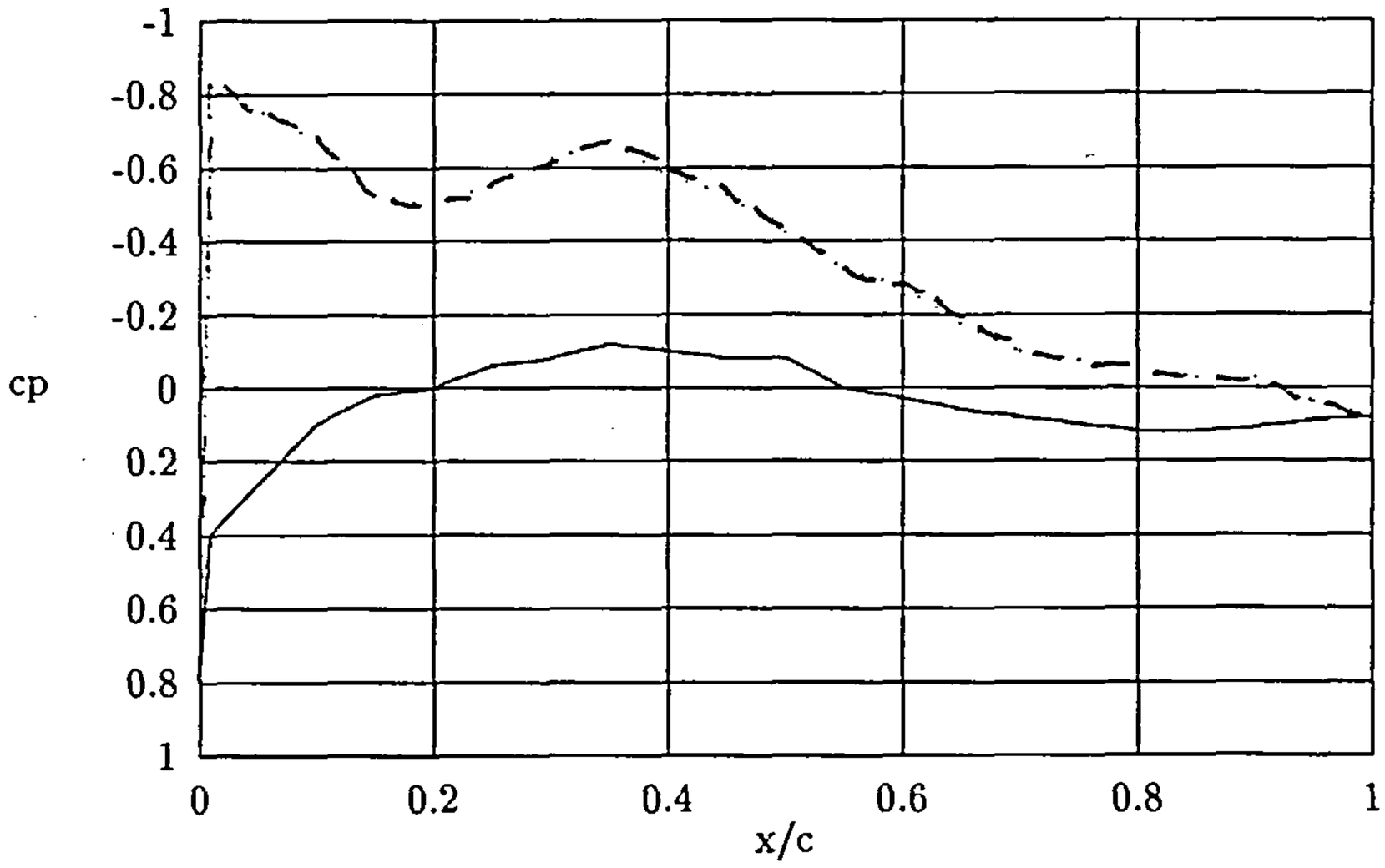

Figure 7.7: c_p distribution at $Y/S = 3$ for optimized fillet

pendix G and frequent reference to them will be made.

7.2 Flow Solver Result Comments

For the following discussion, the term *high drag* will denote the configuration which produced the higher interference drag. The optimum configuration exhibiting the lowest interference drag resulted from the response surface analysis of all experimental run results, and was not a part of the original design table.

Before attempting to explain the behaviour of the Navier-Stokes solutions, it must be pointed out that some discrepancies may be expected if the same configuration was experimentally studied. First of all, caution must be exercised about the number of grid points used. Even though the solver returned realistic pressure distributions (pressures greater than C_{crit} but less

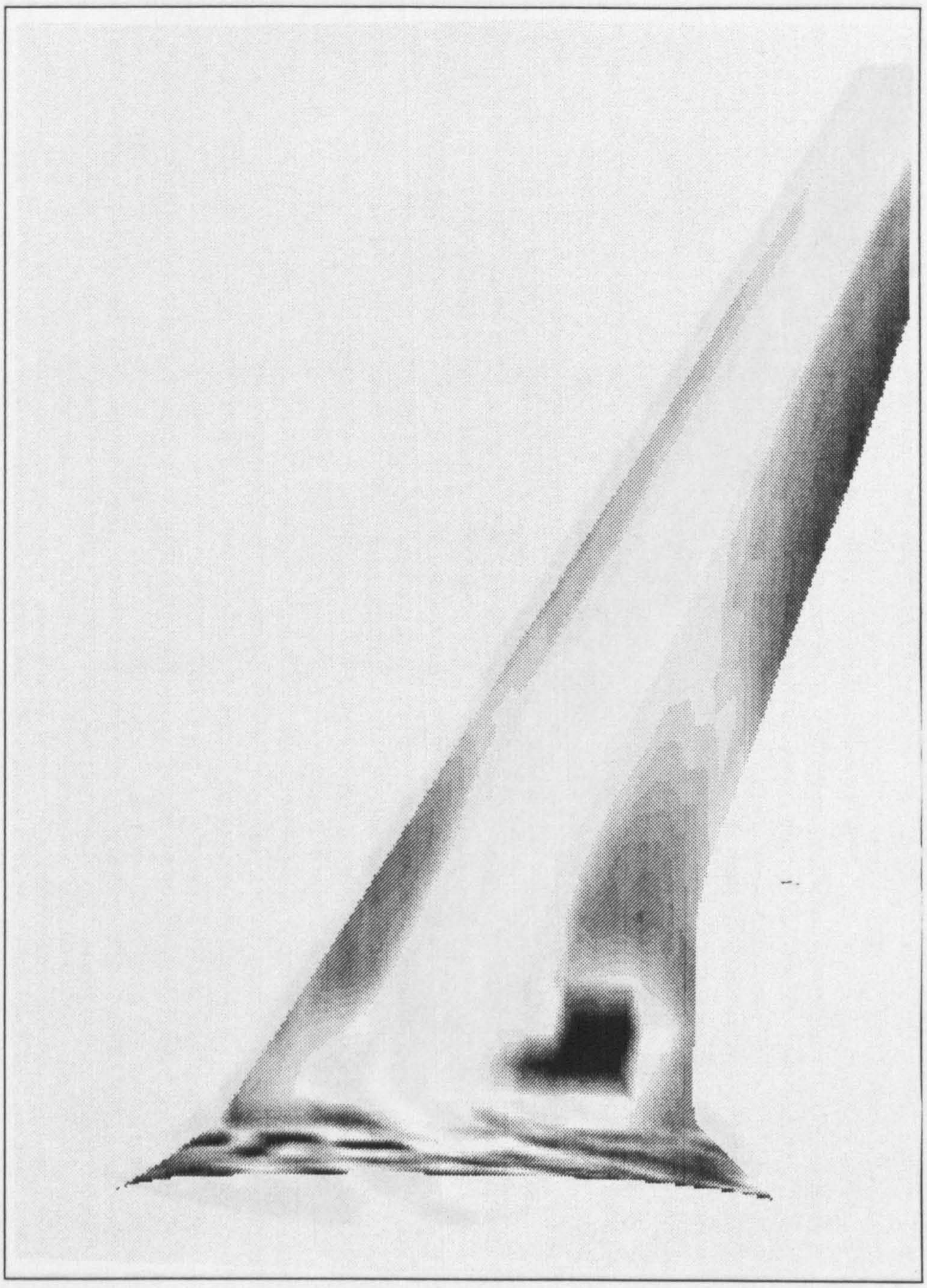

Figure 7.8: Rendered Upper Surface Pressure Distribution of Optimized Fillet

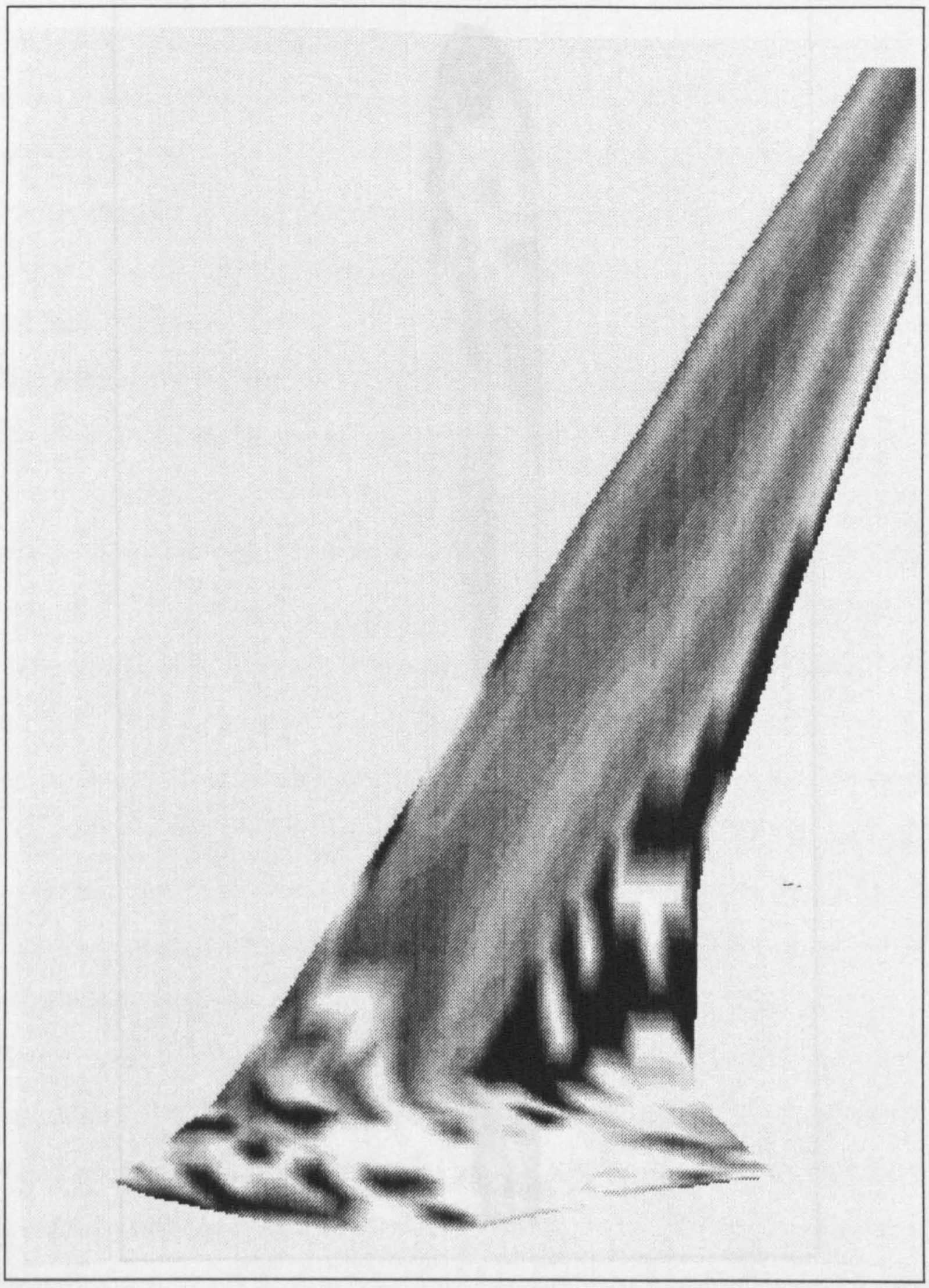

Figure 7.9: Rendered Lower Surface Pressure Distribution for Optimized Fillet

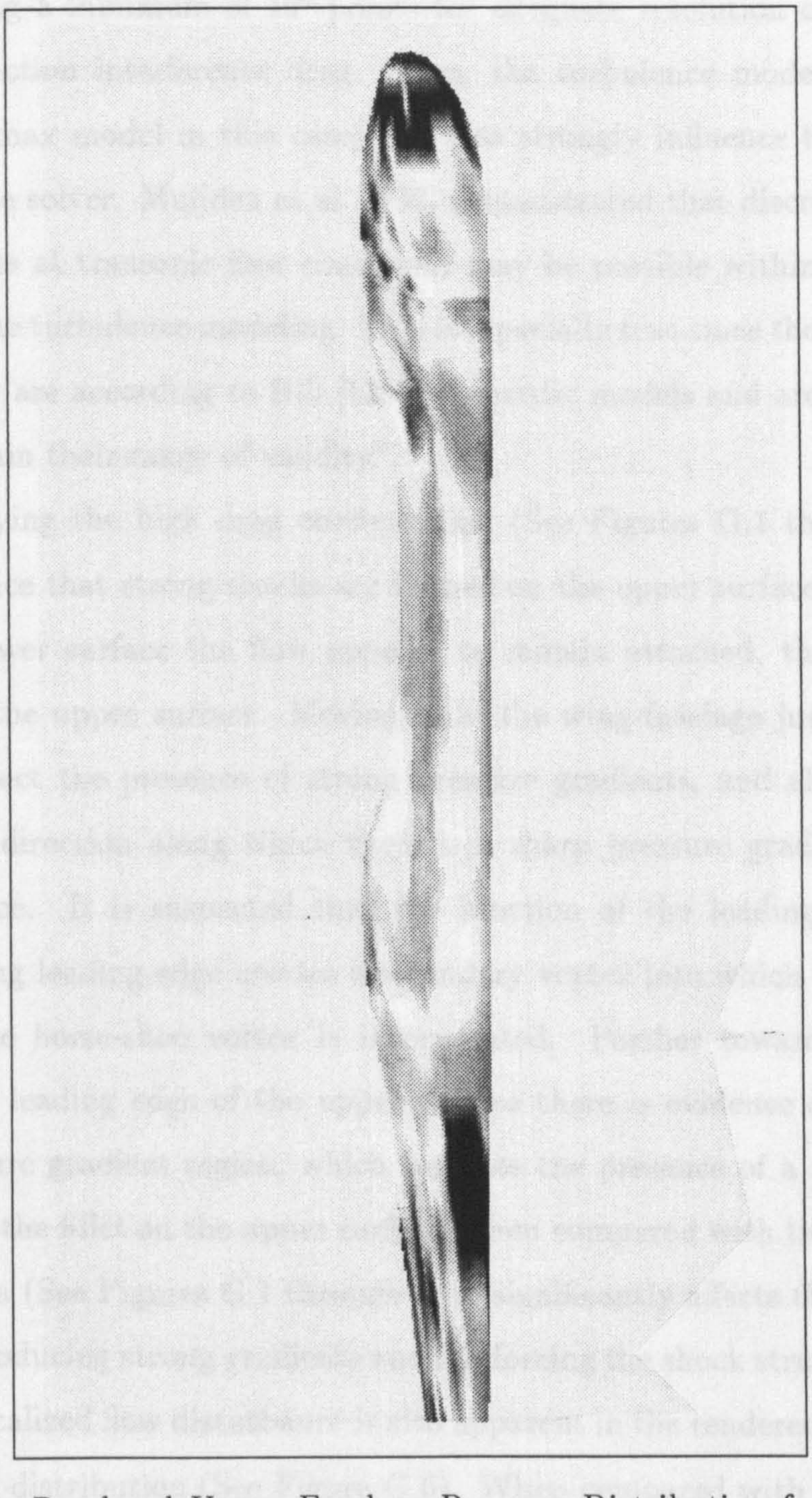

Figure 7.10: Rendered Upper Fuselage Pressure Distribution for Optimized Fillet

than C_{limit}[105][1]) many researchers (e.g. Rill [135] and Chen and Hung[34]) suggest using a minimum of 10^6 points for adequate resolution of the wing-fuselage junction interference drag. Also, the turbulence model used (the Baldwin-Lomax model in this case) can also strongly influence the drag estimate of the solver. Muijden et al. [152] demonstrated that discrepancies up to ten counts at transonic flow conditions may be possible within the uncertainties of the turbulence modeling. This is especially true since the turbulence models used are according to Rill [135]: "heuristic models and are only to be applied within their range of validity".

By studying the high drag configuration (See Figures G.1 through G.5) one can notice that strong shocks are formed on the upper surface. And even if on the lower surface the flow appears to remain attached, this does not happen on the upper surface. Moving in at the wing-fuselage junction area, one can detect the presence of strong pressure gradients, and also a region in the flow direction along which there is a sharp pressure gradient on the upper surface. It is suspected that the junction of the leading edge fillet with the wing leading edge creates a secondary vortex into which the original wing-fuselage horse-shoe vortex is incorporated. Further towards the fillet root, at the leading edge of the upper surface there is evidence of a further sharp pressure gradient region, which suggests the presence of a shock. The geometry of the fillet on the upper surface, when compared with the high drag configuration (See Figures G.1 through G.5) significantly affects the flow over the wing, producing strong gradients and reinforcing the shock strengths. This enhanced localized flow disturbance is also apparent in the rendered pictures of the pressure distribution (See Figure G.6). When compared with Figures 7.3 and 7.4 it can be observed that the flow is more energetic closer to the fuselage in the high drag configuration rather than in the no-fillet configuration.

[1] previous unrealistic results are mentioned in Appendix G

Observing the optimized and high drag fillet pressure distributions, one notices the peculiar offsetting of the suction peak on the upper surface of the wing. A suggestion from Rill [135] is that this is due to the formation of a wing/body junction vortex

Finally, the optimized fillet exhibits behaviour much similar to the previous case, but the flow in this case presents even smoother gradients on the lower surface (See Figures 7.3 to 7.7) possibly attenuating the strength of the shocks. From Figure 7.10 one still detects evidence of the horse-shoe vortex close to the fuselage, while only weak evidence is available on the fillet-wing vortex presence on either upper or lower wing surface. From Figure 7.1 one notices immediately that the shape of the optimized fillet contains a relatively steep surface rise at the front half of the fillet on the upper surface. It is possible that this feature close to the fillet-fuselage intersection curve is able to move the horse-shoe vortex closer to the fuselage, energizing the fuselage boundary layer along with the wing-root region boundary layer and so avoid premature separation as was possibly the case with the high drag configuration. It is also of interest that the leading and trailing edges of the fillet are almost straight lines, while the location of the fillet-wing junction is as far down the span as possible. Finally, the chord scaling factor was at its maximum value while the thickness scaling factor was at its lowest value, indicating that thick fillet geometries possibly promote the migration of the vortex in the spanwise direction while long root-section fillets with appropriate shape distribution aid in "locking" the horse-shoe vortex closer to the junction apparently energizing the local boundary layers and postponing premature flow separation. It is of interest also to notice the fairly sharp leading edge of the fillet. This has been a counter-intuitive development, since most of the fillets used today are either formed of radii or smooth splines. ([86], [33]). However, Bernstein and Hamid recently published a study of a leading edge fillet that is strake-like. The

main effect this peculiar shape had was to reduce significantly the turbulence levels, although they cannot draw any conclusions about any conceivable drag reduction.

7.3 Optimizer Results

It has already been mentioned in Chapter 6 that the optimization method has worked reasonably well with a minimum-point hybrid design which produced a very small (less than 10 percent difference) deviation from the simulated value at the optimum point. The stationary point of the response surface for the six variable simulation was located outside the feasible region, and therefore an optimizer had to be implemented to determine the optimum point within the feasible region. The optimum was found easily and consistently within approximately 20 to 30 function evaluations. The constraints imposed on the optimizer were mainly on the variable bounds, except early in the fillet design where the volume constraints had to be accounted for (the fillet was required to contain a minimum volume for allocation of the retracted landing gear). The variable bounds did not present any problems in the optimization process. After the response surface fit, the objective was well behaved and did not produce significantly sharp ridges around the area of the feasible optimum.

7.4 Optimizer Results Discussion

Very interesting evidence was made apparent with the sensitivity analysis around the optimum region. Two of the design variables were almost non-influential at the optimum as shown by the appropriate sensitivity studies (See Chapter 6). These were the two shape factors for the fillet-wing and the

fillet-fuselage interface. The most important factors were the scaling factors for the thickness and length of the fuselage root-section which seem to strongly influence the behaviour of the horse-shoe vortex and therefore the flow pattern around the area of the junction. Another result of the optimization was that the length of the fillet was kept consistently at the longest value, indicating that a shorter fillet with this particular geometry definition (Bezier control net with 5×4 points and tangency constraints at the bounding curves) contributed to either the migration of the horse-shoe vortex along the span closer to the fillet-wing intersection line, or reinforced the sharp gradients in the junction region.

A further analysis involving more variables would be useful in verifying the above conclusions. Such an analysis involving eleven variables and eighty iterations was attempted earlier, but due to an error in the surface patch description code (the boundary curve tangent directions were taken to be the opposites of the appropriate ones, leading to very localized, intensely warped surfaces), was inconclusive.

Not enough point values were collected during the simulation runs to perform a goodness of fit test for the response surface, but given that the agreement between the predicted and actual optimum values is so high the results are considered to be reliable.

The method presented in this thesis, consisting of a combination of statistical, analytical and computational methods, demonstrated that the preliminary aerodynamic optimization of the wing-fuselage junction fillet is feasible, potentially more efficient and more economical from the methods currently used. Such methods currently consist of classical methods of optimization. These methods being more efficient than stochastic or non-conventional methods (such as Simulated Annealing and Genetic Algorithms) when applied to objective functions that behave smoothly in the region of interest, are not eco-

nomical when it comes to expensive objective function evaluations. The implementation of the Design of Experiments and Response Surface methodologies significantly improved the efficiency of the optimization process by reducing the number of objective evaluations required, and by being able to provide sensitivity information with no extra evaluations. On the other hand, the method should be used with caution, and only after the designer is convinced that the objective function behaves smoothly in the region of interest and for the designer's purposes. Familiarity with the behavior of the objective function is important in the choice of the objective approximation, especially when the number of variables is large. The linear approximation requires a number of evaluations proportional to the number of variables while the quadratic approximation requires a number of evaluations proportional to the square of the number of variables. For a large number of variables, it is important to begin with a linear approximation of the objective using the corner points of a CCD (or similar) design (see Figure 2.2) and then perform a goodness of fit analysis to the linear response surface. If the fit is unacceptable, further runs to obtain values for the star points and the center point of the CCD design (see Figure 2.2) should be performed.

The implementation is also modular, and easily replaceable with a different optimizer-flow solver combination. This modularity was demonstrated by the incorporation of a simple landing gear design module which provided one of the constraints in the original design of the fillets.

Concluding, the method presented in this thesis significantly reduced the effort required for the optimization of the wing-fuselage fillet. The method still depends on approximately the square of the number of variables divided by two and therefore the maximum number of variables that could be used for a realistic implementation of the method relies on the expense of a functional evaluation. If an Euler code were used instead of a Navier-Stokes code, the

functional evaluations would require less time, thus allowing for more variables to be considered. The use of hybrid designs, as mentioned above, improves the efficiency of the method when the designer is convinced that the behavior of the objective can be adequately approximated by a linear or a quadratic response surface. Finally, the method is modular and easy to use, assuming the designer is already familiar with the flow solver and the classical optimization modules.

Chapter 8

Conclusions

8.1 Summary and Conclusions

This thesis suggested a method for incorporating important but computationally intensive modules into the preliminary aircraft design, and a demonstration case of minimizing the interference drag of the wing-fuselage junction via the design of a filleting surface subject to constraints imposed by the user and also interdisciplinary constraints imposed by the landing gear design module. The fillet was described by a set of six variables controlling the behaviour of two sets of five by four control points which in turn fully defined the upper and lower surface of the fillet. The design variables were a combination of purely geometric variables controlling the surface shape and aerodynamic design variables such as wing-root chord, wing-root chord thickness, and span. The demonstration led to the following conclusions:

- The inclusion of the Fillet Optimization Module (FOM) into the preliminary aircraft design is possible. The response surface can provide reliable predictions on the localized behaviour of the flowfield around the junction and predict the optimum parameter values for minimized

interference drag. Implementation of the response surface allows for a large number of predictions for almost no computational cost without having to perform the Navier Stokes Analysis each time.

- The interference drag can be reduced significantly by the application of the appropriate fillet in the wing-fuselage junction. In the demonstration case presented in the thesis, the interference drag was reduced by approximately fifty percent compared to the interference drag without the presence of the fillet.

- It is possible to provide inputs to the FOM from another inter-disciplinary module and even include further design variables into the FOM, but large number of variables increases the number of runs required (for n variables $1/2(n+1)(n+2)$ runs are required). It is suggested therefore to perform the fillet optimization implementing the minimum number of variables, obtain the response surface and use it as part of a larger optimization if required.

- The method of Design of Experiments (DOE) and the Response Surface Methodology (RSM) are capable of modeling adequately the lowest interference drag fillet configurations within the given fillet definition framework.

- Given that at different conditions the interference drag can account up to thirty percent of the sum of the drag forces experienced by the wing and the fuselage, each without the presence of the other, (i.e. during take-off and landing) the reduction of the interference drag is of importance. Therefore, the incorporation of the fillet design into the preliminary aircraft design phase is strongly recommended.

- DOE and RSM can provide powerful tools for preliminary investigation of a high-dimension design space with minimum knowledge of the objective function behaviour. The assumption of local smoothness is valid for most cases in the aerodynamic design field, thus rendering these methods applicable for quick, comprehensive and reliable description of the feasible domain.

The visualization of the results required advanced visualization software. Such software were on a limited availability at Cranfield, and consisted of a software package that was outdated and clumsy to implement (AVS v1.0). Furthermore, this system is soon to be removed from the Computer Center servers, with no replacement on the way. Such a visualization package would be invaluable in advanced flow field analyses.

8.2 Suggestions for Further Work

This thesis demonstrated that for cases that were considered until today too computationally expensive to perform (such as an optimization with a Navier-Stokes flow solver result as the objective function), it is possible to achieve a compromise by obtaining a response surface. This surface would simulate the solver outputs over a limited area of interest in the design space. Some suggestions for the improvement of the efficiency of the method follow, along with suggestions for further work.

- It is essential for the efficiency of the surface and grid generation to implement a graphical interface for the corresponding modules. Such an interface is not currently available for the TLNS code.

- A code providing information on the flow properties at any point of the computational grid would provide a wealth of information about the flow

behaviour around the wing-fuselage junction and would assist greatly in the understanding of the flow behaviour.

- A scientific visualization system is also vital for advanced flow analysis purposes. Such a system is no longer available at Cranfield.

- A greater grid resolution with appropriately higher computational demands could be set up to determine the sensitivity of the results on the grid resolution. Along the same lines, a number of different control functions could be used in the grid generation module to produce different spacings between the points in the computational region. It is conceivable that better resolution and more efficient grid point spacing could accelerate the convergence of the flow solver. Finally, an assessment of the effects of the turbulence model used and its effects on the predicted location and strength of the shocks would provide further insight to the problem.

- For the particular problem of the interference drag, using an Euler code instead of a TLNS code would provide an interesting comparison indicating the instances where the viscous effects are dominant, and outlining (or rejecting) the requirement of a viscous flow solver.

- Further applications of the method include low Reynolds number and low Mach number runs simulating experimental runs. This would possibly not require the TLNS code, and an Euler code should be adequate to obtain some reasonable results which could be validated either by direct wind-tunnel measurements or by flow visualization methods. A surface optimization result could then be tunnel-tested and validated experimentally as well.

- Low speed applications of the method using an appropriate flow solver could have potential applications in many instances of juncture flows in addition to the aeronautical discipline, such as appendage/hull junctures on ships, high-rise buildings, bridge piers in rivers, and the blade passages in turbomachineries. The ability to predict these juncture flows with reasonable accuracy and efficiency is of great practical interest. It is possible that by combining the DOE-RSM minimum point design of a response surface with an elaborate flow solver, the juncture flow analysis and optimization of the above cases can be carried out at the preliminary design stage.

Bibliography

[1] J.M. Allen and C. B. Watson. Experimental effects of wing location on wing–body pressures at supersonic speeds. *NASA TM–4434*, April 1993.

[2] D.A. Anderson, J.C. Tannehill, and R.H. Pletcher. *Computational Fluid Mechanics and Heat Transfer.* Series in Computational Methods in Mechanics and Thermal Sciences. Hemisphere Publishing Corporation, 1984.

[3] P.R. Ashill. CFD methods for drag prediction and analysis currently in use in UK. *RAE TM Aero 2146*, 1988.

[4] H. Ashley and M. Landahl. *Aerodynamics of Wings and Bodies.* Dover Publications, Inc., 1965.

[5] A.C. Atkinson. The design of experiments to estimate the slope of a response surface. *Biometrika*, 57:319–328, 1970.

[6] C.J. Baker. The laminar horseshoe vortex. *Journal of Fluid Mechanics*, 95, 2:347–367, 1979.

[7] C.J. Baker. The turbulent horseshoe vortex. *Journal of Fluid Mechanics*, 6:9-23, 1980.

[8] B.S. Baldwin and H. Lomax. Thin layer approximation and algebraic model for separated turbulent flows. *AIAA, 78-0257*, 1978.

[9] R.L. Barger. A method for designing blended wing-body configurations for low wave drag. *NASA TP-3261*, September 1992.

[10] R.L. Barger. Trajectory fitting in function space with application to analytic modeling of surfaces. *NASA TP-3232*, July 1992.

[11] R.L. Barger and M.S. Adams. Automatic computation of wing-fuselage intersection lines and fillet inserts with fixed-area constraints. *NASA TM-4406*, 1993.

[12] L. Bernstein and S. Hamid. On the effect of a strake-like junction fillet on the lift and drag of a wing. *The Aeronautical Journal, 100*, 992:39–52, 1996.

[13] R. Bodden and R. Edwards. A response-surface optimized tableted reagent for the assay of creatinine. *Clinical Chemistry*, 28:1581–1585, 1982.

[14] W. Boehm. Cubic b-spline curves and surfaces in computer aided geometric design. *Computing*, 19:29–34, 1977.

[15] G.E.P Box. Multifactor designs of first order. *Biometrika*, 39, 1952.

[16] G.E.P Box and D.W. Behnken. Some new three-level designs for the study of quantitative variables. *Technometrics*, 2:455–475, 1960.

[17] G.E.P. Box and N.R. Draper. *Empirical Model-Building and Response Surfaces.* Wiley Series in Probability and Mathematical Statitics. John Wiley and Sons, 1987.

[18] G.E.P. Box and J.S. Hunter. Multi-factor experimental designs for exploring response surfaces. *Annals of Mathematical Statistics*, 28:195–241, November 1957.

[19] G.E.P. Box, W.G. Hunter, and J.S. Hunter. *Statistics for Experimenters: An Introduction to Design, Data Analysis and Model Building.* Wiley Series in Probability and Mathematical Statistics. John Wiley, 1978.

[20] G.E.P Box and K.B. Wilson. On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society*, 13:1–45, 1951.

[21] M.J. Box and N.R. Draper. A basis for the selection of a response surface design. *Journal of the American Statistical Association*, 54:622–654, 1959.

[22] M.J. Box and N.R. Draper. The choice of a second order rotatable design. *Biometrika*, 50:335–352, 1963.

[23] M.J. Box and N.R. Draper. On minimum-point second order designs. *Technometrics, 16*, 4:613–616, November 1974.

[24] G. W. Burgreen and O. Baysal. Three-dimensional aerodynamic shape optimization of wings using sensitivity analysis. *AIAA, 94-0094*, January 1994.

[25] G. W. Burgreen, O. Baysal, and M. E. Eleshaky. Improving the efficiency of aerodynamic shape optimization. *AIAA Journal, 32*, 1:69–76, January 1994.

[26] R.L. Campbell. An approach to constrained aerodynamic design with application to airfoils. *NASA TP-3260*, November 1992.

[27] R.L. Campbell and L.A. Smith. Applications of a direct/iterative design method to complex transonic configurations. *NASA TP-3234*, October 1992.

[28] W.H. Carter, V.M. Vhinvhilli, R.H. Myers, and E.D. Campbell. Confidence intervals and an improved ridge analysis of response surfaces. *Technometrics, 28*, 4:339–346, November 1986.

[29] G.C. Cavanos. Criteria for the optimal design of experimental tests. *NASA TM X-2663*, November 1972.

[30] D.R. Chapman. Computational aerodynamics development and outlook. *AIAA*, 17:1293–1313, 1979.

[31] D.R. Chapman. In 7th international conference for numerical methods in fluid dynamics. In *Lecture Notes in Physics.* 141:1–11, Springer, Berlin,Heidelberg, 1981.

[32] D.R. Chapman, H. Mark, and M.W. Pirtle. Astronautics and aeronautics, 22–23, 1975.

[33] D.C. Cheatham and M.C. Kurbjun. Transonic drag characteristics of a wing–body combination showing the effect of a large wing fillet. *NACA RM L8F08*, 1948.

[34] C.L. Chen and C.M Hung. Numerical study of juncture flows. *AIAA 91-1660*, June 1991.

[35] L.J. Clancy. *Aerodynamics.* Pitman Publishing, 1975.

[36] W.W. Claycomb and W.G. Sullivan. Use of response surface methodology to select a cutting tool to maximize profit. *Journal of Industrial Engineering*, 98:63–65, 1976.

[37] D.B. Close, A.D. Robbins, P.H. Rubin, and R. Stallman. *The GAWK Manual, Version 0.15.* Free Software Foundation, Inc., April 1993.

[38] R. Courant, K.O. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics (translation). *IBM J. Res. Dev*, 11:215–234, 1928.

[39] N.S. Currey. *Aircraft Landing Gear Design: Principles and Practices.* AIAA Education Series. AIAA, 1988.

[40] W. Davenport, M.DeWitz, N. Agarwal, R. Simpson, and K. Podder. Effects of a fillet on the flow past a wing body junction. *AIAA, 89-0986*, 1989.

[41] E.B. Dean. Parametric cost analysis: A design function. In *33rd Annual Meeting of the American Association Cost Engineers*, San Diego CA, June 1989.

[42] E.B. Dean and R. Unal. Designing for cost. In *Conference of the American Association of Cost Engineers*, 1991.

[43] G. Derringer and R. Suich. Simultaneous optimization of several response variables. *Journal of Quality Technology, 12*, 4:214–219, October 1980.

[44] W.J. Diamond. *Practical Experiment Designs for Engineers and Scientists.* Competitive Manufacturing Series. Van Nostrand Reinhold, 2nd edition, 1989.

[45] A.F. Donovan and H.R. Lawrence, editors. *Aerodynamic Components of Aircraft at High Speeds*, volume VII of *High Speed Aerodynamics and Jet Propulsion.* Oxford University Press, 1957.

[46] J. Douglas and H.H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American Mathematical Society*, 82:421–439, 1956.

[47] N.R. Draper. Ridge analysis of response surfaces. *Technometrics, 5*, 4:469–479, November 1963.

[48] N.R. Draper. On lack of fit. *Technometrics, 13*, 2:231–241, May 1971.

[49] N.R. Draper. Center points in second-order response surface designs. *Technometrics, 24*, 2:127–133, May 1982.

[50] A.J. Duncan. *Quality Control and Industrial Applications, Fifth Edition.* IRWIN Publishing, 1986.

[51] W.A. Eckerle and L.S. Langston. Horseshoe vortex formation around a cylinder. *Journal of Turbomachinery*, 109:278–285, April 1987.

[52] W.C. Engelund, D.O. Stanley, R.A. Lepsch, and M.M McMillan. Aerodynamic configuration design using response surface methodology analysis. *AIAA, 93-3967*, August 1993.

[53] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide.* Academic Press, Inc., 2nd edition, 1990.

[54] C. Ferrari. Interaction problems. In A.F. Donovan and H.R. Lawrence, editors, *Aerodynamic Components of Aircraft at High Speeds*, volume IV, Section C of *Princeton Series in High Speed Aerodynamics and Jet Propulsion.* Princeton University Press, 1957.

[55] C.A.J. Fletcher. *Computational Galerkin Methods.* Springer Series in Computational Physics. Springer, Berlin,Heidelberg, 1984.

[56] G.E. Forsythe and W. Wasow. *Finite Difference Methods for Partial Differential Equations.* Wiley, New York, 1960.

[57] R.L. Fox. *Optimization Methods for Engineering Design.* Addison–Wesley Publishing Company, 1971.

[58] S.P. Frankel. Convergence rates of iterative treatments of partial differential equations. *Mathematical Tables and Other Aids to Computation*, 4:635–649, 1950.

[59] B. Gatlin. *EAGLE Basics.* Mississippi State University/National Science Foundation, 1988.

[60] B. Gatlin. *Program EAGLE User's Manual. Volume I: Introduction and Grid Applications.* Air Force Armament Laboratory, 1988.

[61] B. Gatlin. *Program EAGLE User's Manual. Volume II: Surface Generation Code.* Air Force Armament Laboratory, 1988.

[62] B. Gatlin. *Program EAGLE User's Manual. Volume III: Grid Generation Code.* Air Force Armament Laboratory, 1988.

[63] B. Gatlin. *Program EAGLE User's Manual. Volume IV: Multiblock, Implicit, Steady–State Euler Code.* Air Force Armament Laboratory, 1988.

[64] W. Gentzsch and K.W. Neves. Computational fluid dunamics: Algorithms and supercomputers. *NASA*, 1988.

[65] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright. User's guide for NPSOL: A FORTRAN package for nonlinear programming. *Stanford University, Department of Operations Research*, SOL86–2, 1986.

[66] P.W. Glynn. Optimization of stochastic systems. In *Proceedings of the Winter Simulation Conference*, pages 52–59. Washington, D.C., 1986.

[67] J.E. Green. *Numerical Methods in Aeronautical Fluid Dynamics.* ed. P.L. Roe, Academic Press, London, 1982.

[68] C. Haberland, J. Thorbeck, and W Fenske. A computer augmented procedure for commercial aircraft preliminary design and optimization. In *Proceedings of the 14th Conference of the International Council of Aeronautical Sciences*, number ICAS-84-4.8.1, pages 943–953, 1984.

[69] R.J. Hader and S.H. Park. Slope-rotatable central composite designs. *Technometrics*, 20, 1978.

[70] R.W. Hamming. *Numerical Methods for Scientists and Engineers.* McGraw-Hill, New York, 1962.

[71] A.S. Heller, E. Oelkers, and D.A. Farnswor. Response surface methodology applied to thermal-hydraulic margin analysis. *Transactions of the American Nuclear Society*, 26, 1977.

[72] P.A. Henne, editor. *Applied Computational Aerodynamics*, volume 125 of *Progress in Astronautics and Aeronautics.* AIAA, 1990.

[73] C.R. Hicks. *Fundamental Concepts in the Design of Experiments.* Holt, Rinehart and Winston, 1964.

[74] W.J. Hill and W.G. Hunter. A review of surface response methodology: A literature review. *Technometrics*, 8:571–590, 1966.

[75] W.J Hill and W.G. Hunter. Design of experiments for subsets of parameters. *Technometrics*, 16:425–434, 1974.

[76] C. Hirsch. *Numerical Computation of Internal and External Flows: Fundamentals of Numerical Discretization*, volume 1. John Wiley and Sons, 1988.

[77] S.F. Hoerner. *Aerodynamic Drag*, chapter VIII: Interference Drag, pages 8.1–8.19. Diehl Books, 1958.

[78] A.T. Hoke. Economical second-order designs based on irregular fractions of the 3^n factorial. *Technometrics*, 16:375–384, 1974.

[79] M. Holt. *Numerical Methods in Fluid Dynamics.* Springer Series in Computational Physics. Springer, Berlin, Heidelberg, 2nd edition, 1984.

[80] J.S. Hunter and T.H. Naylor. Experimental designs for computer simulation experiments. *Management Science, 16*, 7:422–434, March 1970.

[81] E.J. Ignall. On experimental designs for computer simulation experiments. *Management Science, 18*, 7:384–389, March 1972.

[82] W. E. Milholen II and N. Chokani. Numerical modeling of transonic juncture flow. *AIAA 92-4036*, July 1992.

[83] A. Jameson. Science, 245:361–371, 1989.

[84] E.N. Nilson J.H. Ahlberg and J.L. Walsh. *The theory of splines and their applications.* Academic Press, Inc., 1967.

[85] M.E. Johnson and C.J. Nachtsheim. Some guidelines for constructing exact d-optimal designs on convex design spaces. *Technometrics, 25*, 3:271–277, August 1983.

[86] J. A. Jupp. Interference aspects of the a310 high speed wing configuration. *AGARD CP-285*, 1980.

[87] U.K. Kaul, D. Kwak, and C. Wagner. A computational study of saddle point separation and horseshoe vortex system. *AIAA, 85-0182*, June 1985.

[88] S. B. Kern. Vortex flow control using fillets on a double-delta wing. *Journal of Aircraft, 30*, 6:818–825, Nov.–Dec. 1993.

[89] A.I. Khuri and J.A. Cornell. *Response Surfaces.* Marcel Dekker, New York, 1987.

[90] J.P.C. Kleijnen. *Statistical Techniques in Simulation. Part II.* STATISTICS: Textbooks and Monographs, Volume 9. Marcel Dekker, Inc., 1975.

[91] J.P.C. Kleijnen. Design and analysis of simulations: Practical statistical techniques. *Simulation, 28*, 1977.

[92] E. Krause. Computational fluids, 13:239–269, 1985.

[93] D. Kuchemann. Design of wing junction, fuselage and nacelles to obtain the full benefit of sweptback wings at high Mach numbers. *RAE TM Aero 2219*, 1949.

[94] G. Kuruvila, S. Taasan, and M.D. Salas. Airfoil design and optimization by the one-shot method. *AIAA, 95-0478*, 1995.

[95] A.M. Law and W.D. Kelton. *Simulation Modeling and Analysis.* Industrial Engineering Series. McGraw-Hill International Editions, 2nd edition, 1991.

[96] P.D. Lax. Weak solutions of nonlinear hyperbolic equations and their numerical computation. *Comm. Pure Appl. Math.*, 7:159–193, 1954.

[97] P.D. Lax and B. Wendroff. Systems of conservation laws. *Comm. Pure Appl. Math.*, 13:217–237, 1960.

[98] L.B.Bush, R. Unal, L.F. Rowell, and J.J. Rehder. Weight optimization of an aerobrake structural concept for a lunar transfer vehicle. *NASA TP-3262*, December 1992.

[99] L.R.Kubendran, A. Bar-Sever, and W.D. Harvey. Flow control in a wing/fuselage-type juncture. *AIAA 98-0614*, January 1988.

[100] J.M. Lucas. Optimum composite designs. *Technometrics, 16*, 4:561–567, November 1974.

[101] J.M. Lucas. Which response surface design is best. *Technometrics, 18*, 4:411–417, November 1976.

[102] R.W. MacCormack. The effect of viscosity in hypervelocity impact cratering. *AIAA*, 69-354, 1969.

[103] M.S. Maketon. Optimization simulation: A survey of recent results. In *Proceedings of the 1987 Winter Simulation Conference*, pages 58–67. Atlanta, GA, 1987.

[104] Agusto Martinez. Program eagle–numerical grid generation as applied to advanced airframe configuration. *AIAA, 87-2294*, 1987.

[105] B.W. McCormick. *Aerodynamics, Aeronautics and Flight Mechanics.* John Wiley & Sons, 1979.

[106] J. B. McDevitt and W. M . Haire. Investigation at high subsonic speeds of a body-contouring method for alleviating the adverse interference at the root of a sweptback wing. *NACA RM A54A22*, April 1954.

[107] R. Mead and D.J. Pike. A review of response surface methodology from a biometric viewpoint. *Biometrika*, 31:803–851, 1975.

[108] R.K. Meyer and C.J. Nachtsheim. The coordinate-exchange algorithm for coonstructing exact optimal experimental designs. *Technometrics, 37*, 1:60–69, February 1995.

[109] T.J. Mitchell. An algorithm for the construction of d-optimal experimental designs. *Technometrics, 16*, 2:203–210, May 1974.

[110] T.J. Mitchell and C.K. Bayne. D-optimal fractions of three level variables. *Technometrics*, 20:203–210, 1978.

[111] D.C. Montgomery. *Design and Analysis of Experiments.* John Wiley, 1976.

[112] D.C. Montgomery and V.M. Bettencourt Jr. Multiple response surface methods in computer simulation. *Simulation, 29*, 1977.

[113] M.D. Morris and T.J. Mitchell. Two-level multifactor designs for detecting the presence of interactions. *Technometrics, 25*, 4:345–355, November 1983.

[114] R. Mukerjee and S. Huda. Minimax second-and-third-order designs to estimate the slope of the response surface. *Biometrika*, 72:173–178, 1985.

[115] H. Multhopp. Aerodynamics of the fuselage. *NACA TM-1036*, 1041.

[116] V.N. Murty and W.J. Studden. Optimal designs for estimating the slope of a polynomial regression. *Journal of the American Statistical Association*, 67:869–873, 1972.

[117] R.H. Myers. *Response Surface Methodology. Published by author*, Blacksburg, VA, USA, 1976.

[118] C. J. Nachtsheim. Tools for computer-aided design of experiments. *Journal of Quality Technology, 19*, 3:132–160, July 1987.

[119] G. Nielson. A transfinite, visually continuous, triangular interpolant. In *Geometric Modeling: Algorithms and New Trends*, pages 235–246. SIAM, 1987.

[120] W. Notz. Minimal point second order designs. *Journal of Statistical Planning and Inference*, 6:47–58, 1982.

[121] A.M. De O. Porta Nova and J.R. Wilson. Estimation of multiresponse simulation metamodels using control variates. *Management Science, 35*, 11:1316–1333, November 1989.

[122] G.G. O'Brien, M.A. Hyman, and S. Kaplan. A study of the numerical solution of partial differential equations. *Journal of Mathematical Physics*, 29:223–251, 1950.

[123] L. Olivi. Response surface methodology in risk analysis. In *Synthesis and Analysis Methods for Safety and Reliability Studies*, pages 314–327. Plenum Press, New York, 1980.

[124] J.M. Ortega and R.G. Voigt. SIAM Rev., 27:147–240, 1985.

[125] R. Peyret and T.D. Taylor. *Computational Methods for Fluid Flow.* Springer Series in Computational Physics. Springer, Berlin, Heidelberg, 1983.

[126] F.J. Pierce and I.K. Tree. The mean flow structure on the symmetry plane of a turbulent juncture vortex. *Journal of Fluid Mechanics*, 112:16–20, March 1990.

[127] B. Piper. Visually smooth interpolation with triangular bezier patches. In *Geometric Modeling: Algorithms and New Trends*, pages 221–234. SIAM, 1987.

[128] R.L. Placket and J.P. Burman. The design of optimum multifactor experiments. *Biometrika*, 33, 1946.

[129] M.A. Potsdam, G.A. Intemann, N.T. Frink, R.L. Campbell, L.A. Smith, and S. Pirzadeh. Wing/pylon fillet design using unstructured mesh euler solvers. *AIAA 93-3500*, 1987.

[130] M.Sc. Group Project. M.Sc. group project 1995: Preliminary design of a large commercial transport. *Cranfield University*, 1995.

[131] D.P. Raymer. *Aircraft Design: A Conceptual Approach.* AIAA Education Series. AIAA, 1989.

[132] L.F. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions Royal Society London, Ser. A*, 210:308–357, 1910.

[133] R.D. Richtmyer. *Difference Methods for Initial–Value Problems.* Interscience Publishers, New York, 1957.

[134] R.D. Richtmyer and K.W. Morton. *Difference Methods for Initial Value Problems.* Wiley, New York, 2nd edition, 1967.

[135] S. Rill. 3D Euler and Navier–Stokes simulation tool for design optimization of transport aircraft. In *Proceedings of the 19th Conference of the International Council of Aeronautical Sciences*, number ICAS-94-5.3.1, pages 813–822, 1994.

[136] P.J. Roache. *Computational Fluid Dynamics.* Hermosa, Albuquerque, N.M, 1976.

[137] K.G. Roquemore. Hybrid designs for quadratic response surfaces. *Technometrics, 18*, 4:419–423, November 1976.

[138] J. Roskam. *Airplane Design, Volumes I-VII.* University of Kansas. Roskam Aviation and Engineering Corporation, 1989.

[139] J.S. Rustagi. *Optimization Techniques in Statistics.* Academic Press, 1994.

[140] L.W. Schruben and V.J. Cogliano. An experimental procedure for simulation response surface model identification. In *Communications of the ACM*, number 8 in Volume 30, pages 716–730. Simulation Modeling and Statistical Computing, August 1987.

[141] I.M.M.A. Shabaka and P.Bradshaw. Turbulent flow measurements in an idealized wing/body junction. *AIAA Journal, 19*, 2:131–132, February 1981.

[142] Steven M. Sidik. NAMER : A FORTRAN IV program for use in optimizing designs of two-level factorial experiments given partial prior information. *NASA TN-D-6545*, March 1972.

[143] A. Sigal and E. Lapidot. The aerodynamic characteristics of configurations having bodies with square, rectangular and circular cross-cections at a Mach number of 0.75. *AIAA 87-2429*, 1987.

[144] R.E. Smith and P.A. Kerr. Multidisciplinary analysis of aerospace vehicle design. *AIAA, 92-4773*, 1992.

[145] R.A.E. Wind Tunnel Staff. Design of body-wing junctions for high subsonic M, for swept back wings and symmetrical bodies. *RAE TM Aero 2336*, 1949.

[146] D.M. Steinberg and W.G. Hunter. Experimental design: Review and comment. *Technometrics, 26*, 2:71–115, May 1984.

[147] C. Sung, M.J. Griffin, and D. Taylor. Improvements in incompressible turbulent horseshoe vortex juncture flow calculations. *AIAA 91-0022*, January 1991.

[148] B.G. Tabachnick and L.S. Fidell. *Using Multivariable Statistics.* Harper Collins, 2nd edition, 1989.

[149] F. Teagle. *The LaTeXCookbook*. SERC, 1991.

[150] C. D. Trescot. Effect of streamline contouring in the wing–fuselage juncture in combination with the supersonic area rule on a sweptback wing–fuselage configuration of high fineness ratio. *NASA TN D-387*, 1957.

[151] R. Unal and E.B. Dean. Taguchi approach to design optimization for quality and cost: An overview. In *Conference of the International Society of Parametric Analysis*, 1991.

[152] J. van Muijden, A.J. Broekhuizen, A.J. van der Wees, and J. van der Vooren. Flow analysis and drag prediction for transonic transport transport wing/body configurations using a viscous–inviscid interaction type method. In *Proceedings of the 19th Conference of the International Council of Aeronautical Sciences*, number ICAS-94-2.6.5, pages 1928–1939, 1994.

[153] M.R. Visbal. Numerical investigation of laminar juncture flows. *AIAA, 89-1873*, 1989.

[154] J. von Neumann and R.D. Richtmyer. A method for the numerical calculation of hydrodynamic shocks. *Journal of Applied Physics*, 21:232–237, 1950.

[155] J. Weber and M.G Joyce. Interference problems in wing-fuselage combinations: Part III: Symmetrical swept wing at zero incidence attached to a cylindrical fuselage. *ARC, CP-1333*, 1975.

[156] J. Weber and M.G. Joyce. Interference problems on wing-fuselage combinations: Part IV: The design problem for a lifting swept wing attached to a cylindrical fuselage. *ARC, CP 1334*, 1975.

[157] D.J. Wheeler. *Understanding Industrial Experimentation.* Statistical Process Controls, Inc., 1988.

[158] G. Wichmann. Investigations concerning wing–body interference for compressible flow. In *Proceedings of the 19th Conference of the International Council of Aeronautical Sciences*, number ICAS-94-5.3.2, pages 823–835, 1994.

[159] J.R. Wilson. Future directions in response surface methodology for simulation. In *Proceedings of the 1987 Winter Simulation Conference*, pages 378–381. Atlanta, GA, 1987.

[160] J. Woodwark. *Computing Shape.* Butterworths, 1986.

Appendix A

Geometry Generation Code

The EAGLE surface generation module defines the geometry of the computational block in which the EAGLE grid generation module will construct the computational grid. The surface module can generate certain generic surfaces, or can receive general surfaces from either the grid system or other sources as input for further processing. Curves can also be rotated, stacked or blended to form surfaces. The surfaces, in turn, can be scaled, transformed and concatenated to form general boundary segments that will later be used as inputs to the grid generation module. A number of utility subroutines are available to manipulate curves on surfaces, curve segments and surface segments. The module can also generate curvilinear grids on curved surfaces, in terms of surface parametric coordinates, by interpolation on the splined surface. In addition, the module can generate surface intersections subject to certain conditions. The surface generation code consists of many features allowing the generation of arbitrarily shaped geometries. The following lists a summary of the features currently available:

1. Generation of generic planes conic-section or cubic curves and cubic space curves.

2. Generation of generic conic-section surfaces and cubic surfaces.

3. Generation of surfaces by stacking, rotating, or blending curves.

4. Extraction and concatenation of surface segments.

5. Transformation of surfaces by translation, rotation and scaling.

6. Reversal or switching of point progression on surfaces.

7. Establishment of point distributions by curvature and with specified end, or interior spacings.

8. Establishment of surface parametric grids by transfinite interpolation.

9. Generation of tensor-product surfaces.

10. Generation of surfaces by transfinite interpolation.

11. Generation of grids in curved surfaces.

A.1 Structure of the EAGLE Program

The code operates through a FORTRAN `NAMELIST` input. Each `READ` action of the main program invokes a single operation, e.g. the generation of a surface by a rotating curve, or the definition of a generic surface. General boundary segments are constructed by a sequence of these operations, which are finally stored for later input into the grid generation module. The code can also write the desired curves, surfaces or boundary segments onto files for further processing or plotting.

Before compiling the code, a number of options have to be set, reflecting the system on which the code is run. These options are controlled through a number of `PARAMETER` statements defined in global files. These parameters are:

DIM1,DIM2 Dimensions of the largest surface that can be treated.

DIMSS Maximum number of points that can be stored in core.

DCOR Maximum number of surfaces(curves) that can be stored in core.

DFIL Maximum number of surfaces(curves) that can be stored on file.

DIMV Maximum number of points that can be read for a surface(curve) from the namelist.

DVAL Maximum number of values that can be stored for input as values of quantities on later input statements.

NVALMX Maximum number of terms that can be involved in the calculation of a stored value.

DPNT Maximum number of numbered points that can be used.

A.2 Implementation Overview

After customizing the above variables, the program is ready for use. In the following example, a half-body of an aircraft is defined along with the half-wing and the computational block boundaries required by the grid generation module. The aircraft is described in two computational blocks, one describing the upper wing surface and the upper quarter of the half-fuselage, and the other describing the lower wing surface and the lower quarter of the half-fuselage.

The NAMELIST listing begins with the input of the wing geometry and the fuselage geometry in terms of defining curves along the span and length of the wing and fuselage respectively. These curves are then splined and arranged in such a way that a BLEND operation is possible. This operation will create a surface that joins these curves. This surface, then, will represent the wing and

the fuselage. To illustrate the control over the initial grid input to the grid generation module, a grid is constructed on the surfaces of the fuselage and the wing with a hyperbolic-tangent spacing in the radial and spanwise direction.

When the geometries of the fuselage and the wing are defined and stored in the appropriate core locations, the boundary surfaces are then defined. In this case, the six boundary surfaces of each computational consist of two singularity lines, the fuselage geometry, the plane of symmetry of the half-fuselage, and the common surface the two blocks share between them. The geometric and computational descriptions of the two blocks is shown in Figure 4.8.

A grid based also on hyperbolic tangent spacing is also created on the boundary surfaces, which then are stored on the interface file with the grid generation module. The hyperbolic tangent spacing method has also the added advantage that the grid lines are perpendicular to the surfaces they are attached to, when the spacing is performed on the radial direction.

The storage format is such that a coupling is implemented between the surface description and the grid generation. This implies that the same core storage locations are used in both the surface and grid generation modules without requiring the re-definition of their sizes and properties when used within the grid generation module.

A.3 Surface Module Input File Listing

The current version of the surface generation module exhibits a number of bugs. Knowledge of them may prevent a researcher in the future from attempting to carry out functions that are fully described in the manual but not fully implemented in the code version that is available to Cranfield University.

The most annoying bug of the surface generator is its inability to accept more than two points on a command specifying points for a line creation. It

appears to be an inability of the program to deal with the Unix functionality rather than with the FORTRAN limitations.

The second disturbing feature of this particular version of the surface generator is the intersection of surfaces. It is because of sheer luck that it was possible to obtain curves out of the intersection of two surfaces. Apparently there is a problem with the point spacing on the first surface and to a lesser degree with the point spacing on the second surface. In all the cases that the method was successful, a patch of the second surface had to be made, the point spacing had to be replicated from the first surface to the second surface, and only then it may be possible to carry out the intersection.

Along similar lines, the subtraction of surfaces also depends on the spacing of the "female" surface. The frequent solution is once more to duplicate the number and spacing of points from the "female" surface to the "male" surface and carry out the operation.

Trying to distribute points on a curve or a surface can also be very frustrating. After using the same curve/surface for a number of successive operations, sometimes the program refuses to acknowledge it as a curve/surface any more, and will produce a host of error messages alerting the user. The only solution is to drop the object and recreate it to a new curve id.

```
**************************************************
***
*                Airfoil section 1
**
**************************************************
*         Upper surface
*
$'current',points=2,values=2049.57,3280,0,
                          2667.9,3280,536.307,coreout=10$
$'current',points=2,values=2667.9,3280,536.307,
                          3500.87,3280,754.687,coreout=11$
$'current',points=2,values=3500.87,3280,754.687,
                          4352.83,3280,885.587,coreout=12$
```

```
$'current',points=2,values=4352.83,3280,885.587,
                        5210.25,3280,974.872,coreout=13$
$'current',points=2,values=5210.25,3280,974.872,
                        6070.06,3280,1037.3,coreout=14$
$'current',points=2,values=6070.06,3280,1037.3,
                        6931.08,3280,1080.25,coreout=15$
$'current',points=2,values=6931.08,3280,1080.25,
                        7792.77,3280,1106.56,coreout=16$
$'current',points=2,values=7792.77,3280,1106.56,
                        8654.79,3280,1117.99,coreout=17$
$'current',points=2,values=8654.79,3280,1117.99,
                        9516.88,3280,1115.56,coreout=18$
$'current',points=2,values=9516.88,3280,1115.56,
                        10378.8,3280,1099.24,coreout=19$
$'current',points=2,values=10378.8,3280,1099.24,
                        11240.6,3280,1076.34,coreout=20$
$'current',points=2,values=11240.6,3280,1076.34,
                        12102.2,3280,1047.95,coreout=21$
$'current',points=2,values=12102.2,3280,1047.95,
                        12959.9,3280,961.774,coreout=22$
$'current',points=2,values=12959.9,3280,961.774,
                        13817.3,3280,872.406,coreout=23$
$'current',points=2,values=13817.3,3280,872.406,
                        14670.5,3280,749.321,coreout=24$
$'current',points=2,values=14670.5,3280,749.321,
                        15519.8,3280,601.442,coreout=25$
$'current',points=2,values=15519.8,3280,601.442,
                        16365.1,3280,431.731,coreout=26$
$'current',points=2,values=16365.1,3280,431.731,
                        17206.9,3280,245.995,coreout=27$
$'current',points=2,values=17206.9,3280,245.995,
                        18050,3280,-64.974,coreout=28$
$'concat',corein=10,-28,edge='first',coreout=51$
*$'curdist',corein=51,points=15,coreout=51 $
$'trans',corein=51,origin=0,0,-2220,coreout=51$
*
*  Lower Surface
*
$'current',points=2,values=2049.57,3280,0,

...
```

```
$'concat',corein=30,-48,edge='first',coreout=50$
*$'curdist',corein=50,points=15,coreout=50 $
$'trans',corein=50,origin=0,0,-2220,coreout=50$
*
*****************************************************
***
*                  Airfoil Section 2
...

        Repeat for all airfoil sections

*
*****************************************************
***
*                  Fuselage Section 1
**
*****************************************************
*       Upper Section
*
$'current',points=2,values=-15443,0.09959585922227,1590.25899033380,
           -15443,-138.299462456120,1581.911160924900,coreout=200$
$'current',points=2,values=-15443,-138.299462456120,1581.9111609249,
           -15443,-274.456756647680,1557.091858835200,coreout=201$
$'current',points=2,values=-15443,-274.456756647680,1557.0918588352,
           -15443,-406.249507671610,1516.523708461300,coreout=202$
$'current',points=2,values=-15443,-406.249507671610,1516.5237084613,
           -15443,-531.780757979050,1461.340383614600,coreout=203$
$'current',points=2,values=-15443,-531.780757979050,1461.3403836146,
           -15443,-649.456999652380,1392.992794620400,coreout=204$
$'current',points=2,values=-15443,-649.456999652380,1392.992794620400,
           -15443,-758.031799530320,1313.139509337000,coreout=205$
$'current',points=2,values=-15443,-758.031799530320,1313.1395093370,
           -15443,-856.61523008850,1223.53641893670,coreout=206$
$'current',points=2,values=-15443,-856.61523008850,1223.53641893670,
           -15443,-944.654064197070,1125.938074971700,coreout=207$
$'current',points=2,values=-15443,-944.654064197070,1125.93807497170,
           -15443,-1021.89078957330,1022.01879698120,coreout=208$
$'current',points=2,values=-15443,-1021.89078957330,1022.01879698120,
           -15443,-1088.310385247800,913.316998717750,coreout=209$
$'current',points=2,values=-15443,-1088.310385247800,913.31699871775,
           -15443,-1144.082942246100,801.202288125960,coreout=210$
$'current',points=2,values=-15443,-1144.082942246100,801.20228812596,
```

```
                -15443,-1189.508345491900,686.862296885860,coreout=211$
$'current',points=2,values=-15443,-1189.508345491900,686.86229688586,
                -15443,-1224.967071557600,571.304929083490,coreout=212$
$'current',points=2,values=-15443,-1224.967071557600,571.30492908349,
                -15443,-1250.879197213600,455.37151577610,coreout=213$
$'current',points=2,values=-15443,-1250.879197213600,455.37151557761,
                -15443,-1267.672211548100,339.756839584130,coreout=214$
$'current',points=2,values=-15443,-1267.672211548100,339.75683958413,
                -15443,-1273.543090061700,270.783305420820,coreout=215$
$'current',points=2,values=-15443,-1273.543090061700,270.78330542082,
                -15443,-1276.363295002600,202.238029190810,coreout=216$
$'current',points=2,values=-15443,-1276.363295002600,202.23802919081,
                -15443,-1276.211213419600,134.216014560510,coreout=217$
$'current',points=2,values=-15443,-1276.211213419600,134.21601456051,
                -15443,-1273.1594717248000,66.8034161779030,coreout=218$
$'current',points=2,values=-15443,-1273.1594717248000,66.80341617790,
                -15443,-1267.27458396570000,0.07936776448549,coreout=219$
$'current',points=2,values=-15443,-1267.27458396570000,0.07936776448,
                -15443,-1258.6168097100000,-65.8822701850510,coreout=220$
$'current',points=2,values=-15443,-1258.6168097100000,-65.8822701850,
                -15443,-1247.240199113600,-131.011251896430,coreout=222$
$'current',points=2,values=-15443,-1247.240199113600,-131.0112518964,
                -15443,-1233.192806557000,-195.239382551910,coreout=223$
$'current',points=2,values=-15443,-1233.192806557000,-195.2393825519,
                -15443,-1216.517057942900,-258.499052986760,coreout=224$
$'current',points=2,values=-15443,-1216.517057942900,-258.4990529867,
                -15443,-1197.250260136700,-320.721875976280,coreout=225$
$'current',points=2,values=-15443,-1197.250260136700,-320.7218759762,
                -15443,-1175.425243947500,-381.837427724370,coreout=226$
$'current',points=2,values=-15443,-1175.425243947500,-381.8374277243,
                -15443,-1151.071134345100,-441.772099547480,coreout=227$
$'current',points=2,values=-15443,-1151.071134345100,-441.7720995474,
                -15443,-1124.214243154300,-500.448066862250,coreout=228$
$'current',points=2,values=-15443,-1124.214243154300,-500.4480668622,
                -15443,-1094.879080162600,-557.782385256110,coreout=229$
$'current',points=2,values=-15443,-1094.879080162600,-557.7823852561,
                -15443,-1063.089478293900,-613.686226454110,coreout=230$
$'current',points=2,values=-15443,-1063.089478293900,-613.6862264541,
                -15443,-1028.869827152300,-668.064270167640,coreout=231$
$'current',points=2,values=-15443,-1028.869827152300,-668.0642701676,
                -15443,-992.246406742850,-720.814270861520,coreout=232$
$'current',points=2,values=-15443,-992.246406742850,-720.81427086152,
```

```
            -15443,-953.248809491140,-771.826821094440,coreout=233$
$'current',points=2,values=-15443,-953.248809491140,-771.82682109444,
            -15443,-911.911433813030,-820.985334918150,coreout=234$
$'current',points=2,values=-15443,-911.911433813030,-820.98533491815,
            -15443,-868.275026528830,-868.166275460870,coreout=235$
$'current',points=2,values=-15443,-868.275026528830,-868.16627546087,
            -15443,-822.38824455690,-913.23964984350,coreout=237$
$'concat',corein=200,-220,222,-235,237,edge='first',coreout=253$
$'surdist',corein=253,points=16,coreout=253 $
$'scale',corein=253,scale=1,-1,1,coreout=253$
*
*       Lower Section
*
$'current',points=2,values=-15443,-822.38824455690,-913.23964984350,
          -15443,-774.309198888270,-956.069791559030,coreout=238$
$'current',points=2,values=-15443,-65.320695425189,-1244.90133646150,
          -15443,-0.0780829193840 1,-1246.75930655910000,coreout=251$

...

$'concat',corein=238,-251,edge='first',coreout=252$
$'curdist',corein=252,points=5,coreout=252$
$'scale',corein=252,scale=1,-1,1,coreout=252$
*****************************************************
***
*                 Fuselage Section 2

....
              Repeat for all fuselage sections

$'scale',corein=258,scale=1,-1,1,coreout=258$
*$'trans',corein=258,origin=-1500,0,0,coreout=258$
*
*************************************************************
*****
*       Translate Fuselage Station 4 further downstream
****
*************************************************************
* Correct: $'trans',corein=258,origin=24500,0,0,coreout=260 $
* Correct: $'trans',corein=259,origin=24500,0,0,coreout=261 $
$'trans',corein=258,origin=26500,0,0,coreout=260 $
$'trans',corein=259,origin=26500,0,0,coreout=261 $
```

```
**********************************************************************
**
*       Create the tail section
**
**********************************************************************
$'trans',corein=252,origin=69000,0,2550,coreout=262$
$'trans',corein=253,origin=69000,0,2550,coreout=263$
$'trans',corein=262,scale=1,0,0,origin=3000,0,0,coreout=264$
$'trans',corein=264,origin=0,0,2550,coreout=264$
$'trans',corein=263,scale=1,0,0,origin=3000,0,0,coreout=265$
$'trans',corein=265,origin=0,0,2550,coreout=265$
**********************************************************************
*       Create the Nose Section
**********************************************************************
$'trans',corein=252,scale=1,0.5,0.5,origin=-400,0,0,coreout=266$
$'trans',corein=266,scale=1,0,0,coreout=6$
$'blend',bound=6,266,curves=2,coreout=9$
$'blend',bound=266,252,curves=2,coreout=11$
$'concat',corein=9,11,edge='second',coreout=269$
$'switch',corein=269,reorder='reverse1',coreout=269$
$'trans',corein=253,scale=1,0.5,0.5,origin=-400,0,0,coreout=267$
$'trans',corein=267,scale=1,0,0,coreout=7$
$'blend',bound=7,267,curves=2,coreout=8$
$'blend',bound=267,253,curves=2,coreout=10$
$'concat',corein=8,10,edge='second',coreout=268$
$'switch',corein=268,reorder='reverse1',coreout=268$
**********************************************************************
*
**          Input stage Over
*
**********************************************************************
*
*  Create the fuselage shell
*
*******
*  Upper Fuselage
*******
*$'blend',bound=267,253,curves=3,coreout=268$
$'blend',bound=253,255,curves=3,coreout=271$
$'blend',bound=255,257,distyp='both',
                space=.1,.05,curves=4,coreout=272$
$'blend',bound=257,259,curves=10,coreout=273$
```

```
$'blend',bound=259,261,curves=22,coreout=274$
$'blend',bound=261,263,distyp='both',
                space=.05,.1,curves=11,coreout=275$
$'blend',bound=263,265,curves=3,coreout=276$
$'concat',corein=271,-276,edge='second',coreout=200$
$'trans',corein=200,coreout=599$
$'switch',corein=599,reorder='reverse1',coreout=599$
$'surdist',corein=599,distyp='tanh','both',
                space=.05,space=.05,.15,
                points=16,48,coreout=599$
$'switch',corein=599,reorder='reverse1',coreout=599$
******
*  TESTING FUSELAGE
**********
****
*  Lower Fuselage
*****
*$'blend',bound=266,252,curves=3,coreout=269$
$'blend',bound=252,254,curves=3,coreout=270$
$'blend',bound=254,256,curves=4,coreout=271$
$'blend',bound=256,258,curves=10,distyp='both',
                                space=.4,.1,coreout=272$
$'blend',bound=258,260,curves=22,distyp='both',
                                space=.1,.4,coreout=273$
$'blend',bound=260,262,curves=11,coreout=274$
$'blend',bound=262,264,curves=3,coreout=275$
$'concat',corein=270,-275,edge='second',coreout=201$
$'trans',corein=201,coreout=598$
*$'surdist',corein=598,distyp='linear','interior',
            space=.01,arcint=.5,points=5,48,coreout=598$
**********
*  TESTING FUSELAGE
**********
**********************
*    COMBINE THE TWO FUSELAGES
**********************
$'concat',corein=599,598,edge='first',coreout=597$
$'surdist',corein=597,points=20,48,coreout=597$
*************************************************
*                BLOCK 1
*************************************************
******
```

```
**   Fuselage Intersection with Block1 (Wing Fuselage)
******
**********************************************************
$'trans',corein=51,scale=1.0,1,1.0,coreout=280$
*
*       Correction for the offset by the z-scaling
*                          i.e. (1.9 x 2242)- 2242
*$'trans',corein=280,origin=0,0,2017.8,coreout=280$
$'trans',corein=280,origin=0,-9000,0,coreout=301$
$'trans',corein=280,origin=0,9000,0,coreout=302$
$'blend',bound=301,302,curves=30,coreout=303$
*
*HAH  WAS $'intsec',male='303',female=200,coreout=304$
*
*$'surdist',corein=303,points=20,30,coreout=303$
$'intsec',male=303,female=597,coreout=304$
$'spline',corein=304,coreout=304$
$'curdist',corein=304,points=20,distyp='both',
                                  space=.03,.06,coreout=304$
*
*       Get the end-point value from the intersection line
*
$'getend',corein=304,point='first'$
$'point',point=1$
$'getend',corein=304,point='last'$
$'point',point=2$
*
*HAH
*
* Get upper and lower lines along length of fuselage.  Line 590
*  is the lower line while line 591 is the upper line.
*
$'extracts',corein=599,direct=2,start=16,1,end=16,46,coreout=590$
$'extracts',corein=599,direct=2,start=1,1,end=1,46,coreout=591$
$'switch',corein=590,reorder='switch',coreout=590$
$'switch',corein=591,reorder='switch',coreout=591$
$'spline',corein=591,coreout=591$
*
*  Assign point names to the end-points of the lines
*
$'getend',corein=591,point='first'$
$'point',point=3$
```

```
$'getend',corein=591,point='last'$
$'point',point=4$
$'getend',corein=590,point='last'$
$'point',point=6$
$'getend',corein=590,point='first'$
$'point',point=5$
*
*
$'spline',corein=597,coreout=599$
$'line',r1=5,r2=1,points=13,distyp='both',
                          space=.1,.02,surface='curved',coreout=592$
$'line',r1=2,r2=6,points=15,distyp='tanh',
                          space=.02,surface='curved',coreout=593$
* WAS:      $'concat',corein=592,304,593,coreout=594$
$'concat',corein=592,304,coreout=594$
$'switch',corein=594,reorder='reverse1',coreout=594$
$'switch',corein=593,reorder='reverse1',coreout=593$
$'concat',corein=593,594,coreout=594$
$'switch',corein=594,reorder='reverse1',coreout=594$
*
$'curvmap',male=594,female=591,number=1,coreout=591$
$'spline',corein=597,coreout=597$
$'blend',bound=594,591,curves=16,distyp='tanh',
                          space=.055,surface='curved',coreout=595$
*$'surdist',corein=595,points=48,15,coreout=595$
$'switch',corein=595,reorder='switch',coreout=595$
$'switch',corein=276,reorder='reverse1',coreout=276$
$'concat',corein=268,595,276,edge='second',coreout=595$
**************************************************
*********Upper fuselage finished and stored in 595
**************************************************
**************************************************
$'trans',corein=50,scale=1.0,1,1.0,coreout=380$
$'trans',corein=380,origin=0,-9000,0,coreout=401$
$'trans',corein=380,origin=0,9000,0,coreout=402$
$'blend',bound=401,402,curves=30,coreout=403$
*$'surdist',corein=403,points=20,30,coreout=403$
$'intsec',male=403,female=201,coreout=404$
$'spline',corein=404,coreout=404$
*
*       Get the end-point value from the intersection line
*
```

```
$'getend',corein=404,point='first'$
$'point',point=11$
$'getend',corein=404,point='last'$
$'point',point=12$
*
*  Get upper and lower lines along  length of the lower fuselage.
*  Line 590
*  is the lower line while line 591 is the upper line
*
$'extracts',corein=598,direct=2,start=1,1,end=1,46,coreout=591$
$'extracts',corein=598,direct=2,start=5,1,end=5,46,coreout=590$
$'switch',corein=590,reorder='switch',coreout=590$
$'switch',corein=591,reorder='switch',coreout=591$
$'spline',corein=591,coreout=591$
*
*  Assign point names
*
$'getend',corein=591,point='first'$
$'point',point=13$
$'getend',corein=591,point='last'$
$'point',point=14$
$'getend',corein=590,point='last'$
$'point',point=16$
$'getend',corein=590,point='first'$
$'point',point=15$
*
$'spline',corein=597,coreout=599$
* was r1=13 and r2=11
$'line',r1=5,r2=1,points=13,distyp='both',
                space=.1,.02,surface='curved',coreout=592$
$'line',r1=2,r2=6,points=15,distyp='tanh',
                        space=.02,surface='curved',coreout=593$
$'concat',corein=592,404,593,coreout=401$
$'curvmap',male=594,female=401,number=1,coreout=594$
$'curvmap',male=594,female=590,number=1,coreout=590$
$'spline',corein=597,coreout=597$
$'blend',bound=590,594,curves=5,distyp='tanh',
                        space=.01,surface='curved',coreout=596$
$'switch',corein=596,reorder='switch',coreout=596$
$'switch',corein=275,reorder='reverse1',coreout=275$
$'concat',corein=269,596,275,edge='second',coreout=596$
*****************************************************
```

```
**************************************************
*******Lower Fuselage Finished and stored in 596
**************************************************
**************************************************
$'assemble',corein=304,92,133,174,176,178,180,182,184,
            points=20,distyp='both',
            space=.03,.06,change='yes',coreout=300$
$'switch',corein=300,reorder='switch',coreout=300$
$'surdist',corein=300,points=25,20,distyp='both',
                                  space=.01,.04,coreout=300$
$'switch',corein=300,reorder='switch',coreout=300$
$'assemble',corein=404,91,132,173,175,177,179,181,183,
             points=20,distyp='both',
             space=.03,.06,change='yes',coreout=301$
$'switch',corein=301,reorder='switch',coreout=301$
$'surdist',corein=301,points=25,20,distyp='both',
                                  space=.01,.04,coreout=301$
$'switch',corein=301,reorder='switch',coreout=301$
**************************************************
*       WING SURFACE DESCRIPTION FINISHED AND STORED IN:
*                UPPER WING  CORE 300
*                LOWER WING  CORE 301
*================================================
*   BEGIN THE WING EXTENSION
*================================================
*       Extract the Wing leading edge
$'extracts',corein=300,direct=2,start=1,1,end=1,25,coreout=321$
*       Extract the Wing Trailing edge
$'extracts',corein=300,direct=2,
                         start=20,1,end=20,25,coreout=322$
*       Get the edge points
$'switch',corein=321,reorder='switch',coreout=321$
$'switch',corein=322,reorder='switch',coreout=322$
*
$'getend',corein=321,point='last'$
$'point',point=21$
$'getend',corein=322,point='last'$
$'point',point=22$
*
$'line',r1=21,r2=44042.3,70000,5095,points=11,coreout=310$
$'line',r1=22,r2=47642,70000,5101.45,points=11,coreout=311$
$'getend',corein=310,point='last'$
```

```
$'point',point=23$
$'getend',corein=311,point='last'$
$'point',point=24$
$'blend',bound=310,311,curves=20,distyp='both',
                                  space=.01,.04,coreout=325$
$'switch',corein=325,reorder='switch',coreout=325$
$'concat',corein=300,325,edge='second',coreout=330$
$'concat',corein=301,325,edge='second',coreout=430$
*-=-----------------------------------------------
*  WING EXTENSION FOR UPPER BLOCK IS STORED IN CORE 330.
*                AND FOR THE LOWER BLOCK IN CORE 430
*-----------------------------------------------
*        Begin the singularity construction lines.
*-----------------------------------------------
*    Leading Edge Singularity.
*    First find the starting point, i.e. the nose.
*
*  Get extended wing spacing by extracting  leading edge
*
$'extracts',corein=330,direct=2,
                  start=1,1,end=1,35,coreout=326$
*
$'getend',corein=596,point=1,1$
$'point',point=31$
$'getend',corein=596,point=1,50$
$'point',point=32$
*  t.e. side
$'scurve',points=17,r2=200000,0,2550,r1=24,t1=1,0,0,
                t2=0,-1,0,distyp='both',
                space=.005,.09,coreout=328$
*  l.e. side
$'scurve',points=15,r1=-200000,0,0,r2=23,t1=0,1,0,t2=1,0,0,
                distyp='both',space=.09,.003,coreout=327$
$'extracts',corein=330,direct=1,
                 start=1,35,end=20,35,coreout=329$
$'concat',corein=327,329,328,coreout=331$
*-----------------
*
*
*
**********************************************
*****SURFACE 6 for UPPER SURFACE  BLOCK IS STORED IN 333
```

```
*****          AND FOR THE LOWER SURFACE IN 334
*****************************************************
*
$'getend',corein=331,point='first'$
$'point',point=33$
$'getend',corein=331,point='last'$
$'point',point=34$
$'line',r2=33,r1=34,points=10,coreout=332$
$'curvec',corein=332,points=10,origin=-31$
$'trans',corein=331,origin=200000,0,0,coreout=331$
$'rotate',bound=331,angpts=16,angle=0,90,
                axcos=0.9999796803069,0,0.006374870462,
                coreout=333$
$'rotate',bound=331,angpts=5,angle=-90,0,
                axcos=0.9999796803069,0,0.006374870462,
                coreout=334$
$'trans',corein=331,origin=-200000,0,0,coreout=331$
$'trans',corein=333,origin=-200000,0,0,coreout=333$
$'trans',corein=334,origin=-200000,0,0,coreout=334$
*****************************************************
**********     GO FOR SURFACE 5 of THE UPPER BLOCK
*****************************************************
$'extracts',corein=333,direct=1,
                start=1,16,end=50,16,coreout=335$
$'extracts',corein=595,direct=2,
                start=16,1,end=16,50,coreout=591$
$'switch',corein=591,reorder='switch',coreout=591$
$'blend',bound=335,591,curves=35,coreout=336$
*****************************************************
*****  UPPER BLOCK SURFACE 5 IS STORED IN CORE 336
*****************************************************
$'extracts',corein=596,direct=2,
                        start=1,1,end=1,50,coreout=591$
$'switch',corein=591,reorder='switch',coreout=591$
$'extracts',corein=334,direct=1,start=1,1,end=50,1,coreout=337$
$'blend',bound=337,591,curves=35,coreout=338$
*****************************************************
**********LOWER BLOCK SURFACE 5 IS STORED IN CORE 338
*****************************************************
*                  begin construction of surface 4
* (common for both blocks, except on the wing surfaces)
*
```

```
*---------------------
*    For the l.e. of the portion of surface 4, do a
*    transfine interpolation with the
*    four curves:one from the front fuselage,
*    one from the extended wing
*    leading edge, one from the freestream boundary,
*    and one from the
*    singularity line.
*
*           Extract fuselage line
$'extracts',corein=595,direct=2,start=1,1,end=1,15,coreout=340$
*           Extract extended wing leading edge
$'extracts',corein=330,direct=2,start=1,1,end=1,35,coreout=341$
*           Core 327 contains the boundary curve
*           Extract Singularity Line
$'extracts',corein=336,direct=2,start=1,1,end=1,35,coreout=342$
$'switch',corein=341,reorder='switch',coreout=341$
$'switch',corein=341,reorder='reverse1',coreout=341$
$'switch',corein=342,reorder='switch',coreout=342$
$'switch',corein=340,reorder='switch',coreout=340$
$'transur',upper1=340,upper2=341,lower1=327,
                                    lower2=342,coreout=345$
*****************************************
******  The front portion of surface 4 is stored in core 345
*****************************************
*******
********Repeat for the rear portion of surface 4
*****************************************
*       Extract fuselage line
$'extracts',corein=595,direct=2,start=1,34,end=1,50,coreout=350$
*       Extract Extended wing trailing edge
$'extracts',corein=330,direct=2,start=20,1,end=20,35,coreout=351$
*       Core 328 contains the boundary curve
*       Extract singularity line
$'extracts',corein=336,direct=2,start=50,1,end=50,35,coreout=352$
$'switch',corein=351,reorder='switch',coreout=351$
$'switch',corein=351,reorder='reverse1',coreout=351$
$'switch',corein=352,reorder='switch',coreout=352$
$'switch',corein=350,reorder='switch',coreout=350$
$'transur',upper1=350,upper2=352,
                              lower1=328,lower2=351,coreout=355$
*****************************************
```

```
*******   The rear portion of surface 4 is stored in core 355$
*****************************************
*******   Begin final assembly of the two surfaces.
*******    Upper block first
*****************************************
$'switch',corein=330,reorder='switch',coreout=330$
$'switch',corein=330,reorder='reverse1',coreout=330$
$'concat',corein=345,330,355,edge='second',coreout=360$
*****************************************
********UPPER BLOCK SURFACE 4 STORED IN CORE 360
*****************************************
******repeat for lower block
*****************************************
$'switch',corein=430,reorder='switch',coreout=430$
$'switch',corein=430,reorder='reverse1',coreout=430$
$'concat',corein=345,430,355,edge='second',coreout=370$
*****************************************
*******LOWER BLOCK SURFACE 4 STORED IN CORE 370
*****************************************
*****   CREATE NOSE SINGULARITY IN CORE 380 FOR UPPER BLOCK
*****                                   381 FOR LOWER BLOCK
*****************************************
$'blend',bound=342,342,curves=16,coreout=380$
$'blend',bound=342,342,curves=5,coreout=381$
*****************************************
*****   CREATE TAIL SINGULARITY IN CORE 390 FOR UPPER BLOCK
*****                                   391 FOR LOWER BLOCK
*****************************************
$'blend',bound=352,352,curves=16,coreout=390$
$'blend',bound=352,352,curves=5,coreout=391$
*_______________________________________
*****************************************
*****    PRINTING COMMANDS
*****************************************
*   PREPARE SEGMENTS FOR PRINTING.  SEGMENTS OF THE UPPER BLOCK
*****   APPER AS 10X WHILE SEGMENTS OF THE LOWER BLOCK APPEAR AS
*****   15X.
*****************************************
$'current',corein=380,coreout=101$
$'current',corein=595,coreout=102$
$'current',corein=390,coreout=103$
$'current',corein=360,coreout=104$
```

```
$'current',corein=336,coreout=105$
$'current',corein=333,coreout=106$
*
$'current',corein=381,coreout=151$
$'current',corein=596,coreout=152$
$'current',corein=391,coreout=153$
$'current',corein=370,coreout=154$
$'current',corein=338,coreout=155$
$'current',corein=334,coreout=156$
*
**************************************
*   DO APPROPRIATE ADJUSTMENTS
**************************************
$'switch',corein=101,reorder='switch',coreout=101$
$'switch',corein=101,reorder='reverse2',coreout=101$
$'switch',corein=102,reorder='reverse1',coreout=102$
$'switch',corein=103,reorder='switch',coreout=103$
$'switch',corein=103,reorder='reverse2',coreout=103$
$'switch',corein=104,reorder='switch',coreout=104$
$'switch',corein=104,reorder='reverse2',coreout=104$
$'switch',corein=105,reorder='reverse2',coreout=105$
$'switch',corein=106,reorder='switch',coreout=106$
$'switch',corein=106,reorder='reverse1',coreout=106$
*
*  ADJUSTMENTS FOR BLOCK 2
*
$'switch',corein=151,reorder='switch',coreout=151$
$'switch',corein=151,reorder='reverse2',coreout=151$
$'switch',corein=152,reorder='reverse1',coreout=152$
$'switch',corein=152,reorder='reverse1',coreout=152$
$'switch',corein=153,reorder='switch',coreout=153$
$'switch',corein=153,reorder='reverse2',coreout=153$
$'switch',corein=154,reorder='switch',coreout=154$
$'switch',corein=154,reorder='reverse2',coreout=154$
$'switch',corein=155,reorder='reverse2',coreout=155$
$'switch',corein=156,reorder='switch',coreout=156$
*
*
*
$'combine',content='yes',corein=101,-106,151,-156,
                                    fileout=10,form='list'$
$'end'$
```

Appendix B

Grid Generation Code

The Grid Generation Module of EAGLE is a composite (multi-block) algebraic or elliptic grid generation system designed to discretize the domain in or around any arbitrarily-shaped three-dimensional region. This code combines a three-dimensional, boundary conforming surface generation scheme with a multi-block, three-dimensional elliptic grid generation scheme. The grid generation module receives the cartesian coordinates of the boundary segments generated by the surface generation module and creates the grid within the physical region defined by these boundaries. The grid is created with a number of different control functions, i.e. functions that control the smoothness of the grid, the localized or not effects of local grid distortions, the behaviour of the grid around the boundary conditions, etc. These control functions allow for greater flexibility in the grid design, but use of such functions other than the default ones are beyond the scope of this project.

B.1 Grid Module Structure

B.2 Grid Structure

The grid is constructed of six-sided blocks that fit together to fill the entire physical region. In this case there are two blocks that were defined in the surface generation module and describe the surface of a half-fuselage with a half-wing and the physical space in which they are located. These blocks have, of course, curved sides in the physical region, but correspond to rectangular blocks in the computational region. Each block is of different size, since the only requirement imposed on the definition of the blocks is that they fit together to fill the physical region.

The boundary segment numbers assigned in the boundary generation module were transferred to the grid generation module, although it was not required. This, nevertheless, allowed changes in the number of points on the various segments to be accomplished through the least changes (sometimes even no changes) to the NAMELIST input formats to the geometry and grid generation modules.

The following NAMELIST runstream corresponds to the coupling of the boundary code with the grid code. The only file read contains the core locations of the segments already defined before, and their sizes. To take full advantage of the coupling between the two codes, indirect addressing of points is implemented. This involves the depiction of the corners of the computational blocks by referring to the boundary segments already stored in the file read in the beginning of the runstream. The actual location of the points in the block is set by a series of operation statements, with the first of which defining the first point in an arbitrary manner. This point is usually the $(1, 1, 1)$ point, although, it can be any point in the block. The other numbered points are

then set in relation to previously set points.

The point assignment statements appear for each block, so points common to more than one block will have the same number in each block, but will be assigned appropriate different indices in each block.

After the corners of the blocks are defined, the boundary segments must be located in the blocks. This can be done by appropriate operations that require the starting and ending point of the segment boundaries. These points of course are indirectly referenced by the points defined in the initial part of the runstream. Once more it is possible to use the same segment number in multiple blocks, although it was not required in this case.

The algebraic grid produced thus far is used as the baseline for the development of the elliptic grid. A total of twenty iterations are dictated to allow for the grid to adjust in the most irregular areas of the physical domain. Finally the grid is printed to a file, ready to be used as input to the Thin Layer Navier Stokes solver module.

B.3 Grid Code Input Listing

```
$'store',all='yes',kstore='file',file=21,form='e',
                order=1,2,3,rorder=1,2,3,filnam='fingrid.dat'$
$ 'file', file=20, all='yes',form='list',accpar='optimum',
                itmax=20,tol=1.05E-06,outer='no' $
c
$'block'$
c
*
**********************************************
*       BLOCK 1 DECLARATIONS
**********************************************
* point SETTINGS  seg numbers switched
**********************************************
$'point',point=101,locat=1,1,1$
$'point',point=102,opoint=101,segment=101,direct=+1,ndex=1$
```

```
$'point',point=103,opoint=102,segment=101,direct=+3,ndex=2$
$'point',point=104,opoint=101,segment=101,direct=+3,ndex=2$
$'point',point=105,opoint=101,segment=102,direct=+2,ndex=2$
$'point',point=106,opoint=105,segment=103,direct=+1,ndex=1$
$'point',point=107,opoint=106,segment=103,direct=+3,ndex=2$
$'point',point=108,opoint=107,segment=103,direct=-1,ndex=1$
c
$'size',size=107$
c
c
ccccccccccccccccccccccccccccc
c segment SETTINGS  for surfaces
ccccccccccccccccccccccccccccc
c
$'segment',segment=101,start=101,end=103,class='fix'$
$'segment',segment=102,start=101,end=106,class='fix'$
$'segment',segment=103,start=105,end=107,class='fix'$
$'segment',segment=104,start=102,end=107$
$'segment',segment=105,start=101,end=108,class='fix'$
$'segment',segment=106,start=104,end=107,class='fix'$
**********************************************************
$'block'$
*
**********************************************************
c       BLOCK 2 DECLARATIONS
**********************************************************
* point SETTINGS  seg numbers switched
**********************************************************
$'point',point=101,locat=1,1,1$
$'point',point=102,opoint=101,segment=151,direct=+1,ndex=1$
$'point',point=103,opoint=102,segment=151,direct=+3,ndex=2$
$'point',point=104,opoint=101,segment=151,direct=+3,ndex=2$
$'point',point=105,opoint=101,segment=152,direct=+2,ndex=2$
$'point',point=106,opoint=105,segment=153,direct=+1,ndex=1$
$'point',point=107,opoint=106,segment=153,direct=+3,ndex=2$
$'point',point=108,opoint=107,segment=153,direct=-1,ndex=1$
c
$'size',size=107$
c
c
ccccccccccccccccccccccccccccc
c segment SETTINGS  for surfaces
```

```
ccccccccccccccccccccccccccccccc
c
$'segment',segment=151,start=101,end=103,class='fix'$
$'segment',segment=152,start=101,end=106 ,class='fix'$
$'segment',segment=153,start=105,end=107,class='fix'$
$'segment',segment=154,start=102,end=107$
$'segment',segment=155,start=101,end=108,class='fix'$
$'segment',segment=156,start=104,end=107,class='fix'$
*
*
$'cut',block=1,start=102,end=107,iblock=2,istart=102,iend=107$
*
$'end'$
$'error',blkerr='yes'$
$'system',all='yes'$
$'end'$
```

Appendix C

The Bezier–Bernstein Fillet Patches

The following methods provide for a consistent model throughout the analysis that is then easily modified for optimization.

C.1 Surface Parameterization

The general requirements for any surface parameterization maintain that the shape representations for any shape elements must be sought so that they are not only well behaved in the mathematical sense, but also behave predictably in response to actions by the user. Also, the representation should not be too expensive to compute for any of the purposes for which it may be required. Several methods have been used for describing complex-surfaced objects by means of controllable shape elements. They are based on two dimensional methods for complex curve representations. The two main methodologies for parametric curves are the Bezier splines and the B-splines.

The Bezier curve is a method of formulating a parametric polynomial to meet the constraints of a designer's conceptions instead of the rigid specifi-

cations of interpolation. The technique is based on the input of a series of points forming a "track". The resulting curve starts at the first point of the track and finishes at the last. In addition, the curve starts and finishes with the slopes of the first and last track lines respectively. However, the resulting curve is guaranteed to be smoother than the track shape, due to the *variation diminishing* property of the Bezier curves, and so things cannot get out of hand as they can with classical interpolation.

To generate a Bezier curve, a track of $m+1$ points is used, $\mathbf{b}_{i(i=0,m)}$ each point being a vector quantity and comprises terms for both the x and y coordinates of the point. The curve $\mathbf{Q}(t)$ is then given by:

$$\mathbf{Q}(t) = \sum_{j=0}^{j=m} \frac{m!}{(m-1)!i!} t^i (1-t)^{m-i} \mathbf{b}_i \qquad \text{(C.1)}$$

The weighting function's behaviour at $t = 0$ and $t = 1$ indicates how the merger of the curve and the track is achieved at the first and last points. At each end only the corresponding end track point has any weight, which achieves the positional constraint. On the other track points, only the weighting function of the next adjacent track point leaves the end with anything other than zero slope, and this produces the tangency of track and curve. This is evident by a comparison of the weighting function's behaviour before and after it is differentiated with respect to t. In addition, the weights add up to unity which preserves the curve's independence from the coordinate axes. An example of a cubic Bezier curve corresponding to four particular vertices is shown in Figure C.1.

For more complex curves, splines are recommended. Splines are curves composed of more than one segments [84].

If a large surface is to be constructed, it is unlikely that a single equation will be able to describe it. This is analogous to the need to use splines instead of a single Bezier curve representation of some complicated curve. Similarly,

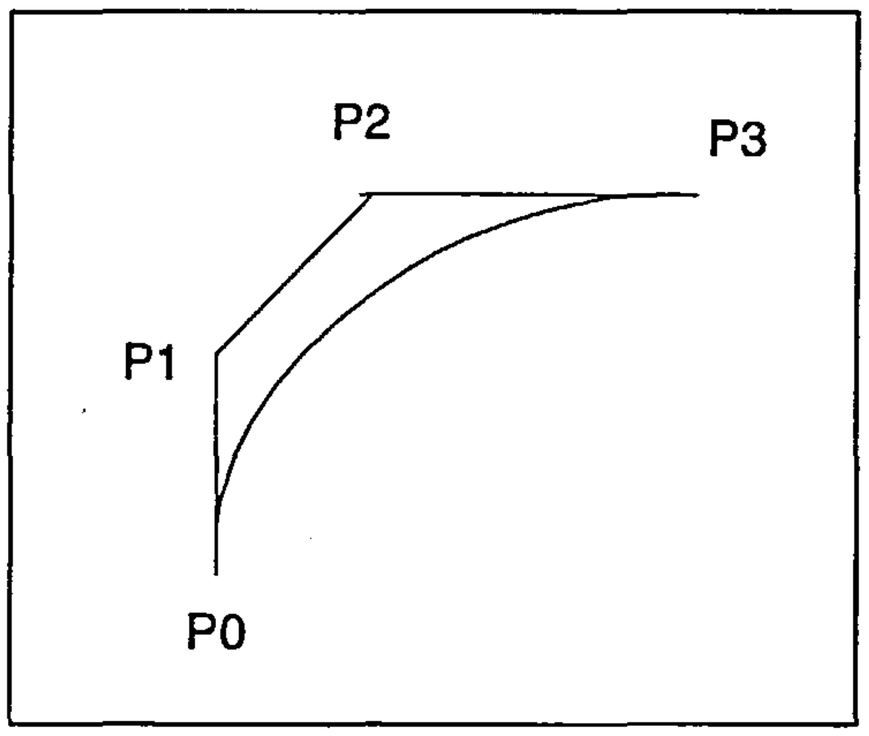

Figure C.1: Example of a Cubic Bezier Curve

Bezier surface patches can be used to represent complicated surface geometries. These patches are formulated in a controllable way which also facilitates joining them together without discontinuities.

The Bezier patch has the equation:

$$\mathbf{Q}(t,u) = \sum_{i=1}^{i=m} \sum_{j=0}^{j=n} \frac{m!}{(m-i)!i!} \frac{n!}{(n-j)!j!} t^i (1-t)^{m-i} u^j (1-u)^{n-j} \mathbf{b}_{i,j} \qquad \text{(C.2)}$$

where the similarity with the Bezier curve formula is apparent. In the Bezier patch the control points are arranged as a grid over the surface. For a bicubic Bezier patch, for instance, a four by four grid of points is needed. The outermost 12 grid points control the boundary curves. The four points on the inside of the grid control the shape of the surface. As was the case with the Bezier curves, the Bezier patch formulation is equivalent to the Cartesian product [160]. Only the method of interpolation is different. Bezier patches can be joined together with both positional and slope continuity by enforcing conditions on the two rows of points in the mesh which correspond to the side of the patch where the joint is to be made. As more patches are added into a surface, the proportion of mesh points that are not predetermined rapidly declines. The only way to obtain more flexibility is to use higher–order patches. With

Bezier patches the number of grid points rises as the square of the order of the edge polynomial. The grids can thus become very computationally intensive, although only the points away from the edge of each grid are freed for the user to specify. Grids of order greater than ten are too computationally intensive to be of any practical use [160].

C.2 Implementation

The Bezier curves mentioned above are given in the Bernstein form, that is they are defined explicitly rather than recursively as they were historically developed by Casteljau [14] in 1959. The Bezier-Bernstein form facilitates considerably the development of the module.

It must be mentioned at this point, that the order of the Bezier-Bernstein polynomials used is cubic, since cubics are the simplest *space curves*, i.e. they are not planar. For two dimensional shapes, piecewise quadratics may suffice, but when it comes to three dimensions they can only produce piecewise two dimensional segments. Another advantage of piecewise cubics is the fact that they may have inflection points inside a segment. With piecewise quadratics, one would have to make sure there is a junction point at every inflection point.

As mentioned in the previous section, it is easy to construct a Bezier curve given the control points, but to achieve the reverse some more work is involved. In this thesis, two methods were investigated: The first involved cubic interpolation, while the second involved the implementation of a higher degree Bezier curve. The main tradeoffs between the two methods lie in the curve description complexity and numerical calculation intensity, the flexibility in the description of arbitrarily smooth geometries and finally the *goodness* of the approximation of the bounding Bezier curves to the original curves to be approximated. For the purposes of this thesis, the cubic Bezier curves are

considered to be too limiting to be exclusively used, and therefore we focus our attention to a combination of cubic and higher order Bezier curves. ion is equivalent to the Cartesian product [160]. Only the method of interpolation is different. Bezier patches can be joined t

C.2.1 Higher Degree Bezier Curves

Bezier curves of higher order than cubic curves are not usually used because of possible incompatibilities with a number of existing Computer Aided Design (CAD) packages, most of which support cubic Bezier curves as the simplest space curves. Nevertheless, the simplicity that a simple higher degree curve offers compared to a cubic spline cannot be ignored. In this case, the higher the degree of the curve, the better the approximation to the curve. Based on the geometry of the wing–fuselage junction, a fourth degree Bezier curve was chosen to represent the fillet–fuselage junction curve for each of the upper and lower wing surfaces, and the fillet–wing junction curve again for each of the upper and lower wing surfaces.

As in the previous section, the main problem that arises is the representation of a given curve by a quartic Bezier curve. An added complication in this case is that the curve to be approximated is defined by a number of points that cannot be readily identified with their corresponding Bezier curve parameter values (parameter t in Eq. C.1). Therefore, a method was devised for estimating the position of the Bezier points without having to obtain a direct equivalence between the parameter t and the points comprising the curve to be approximated. Thus, an optimization method was required, similar to the least squares method. The latter could not be used because of the above inability to associate t-values with the points on the curve to be approximated.

One way to solve the above problem is the following. By creating a surface bounded by the two curves (i.e. the curve to be approximated and the Bezier

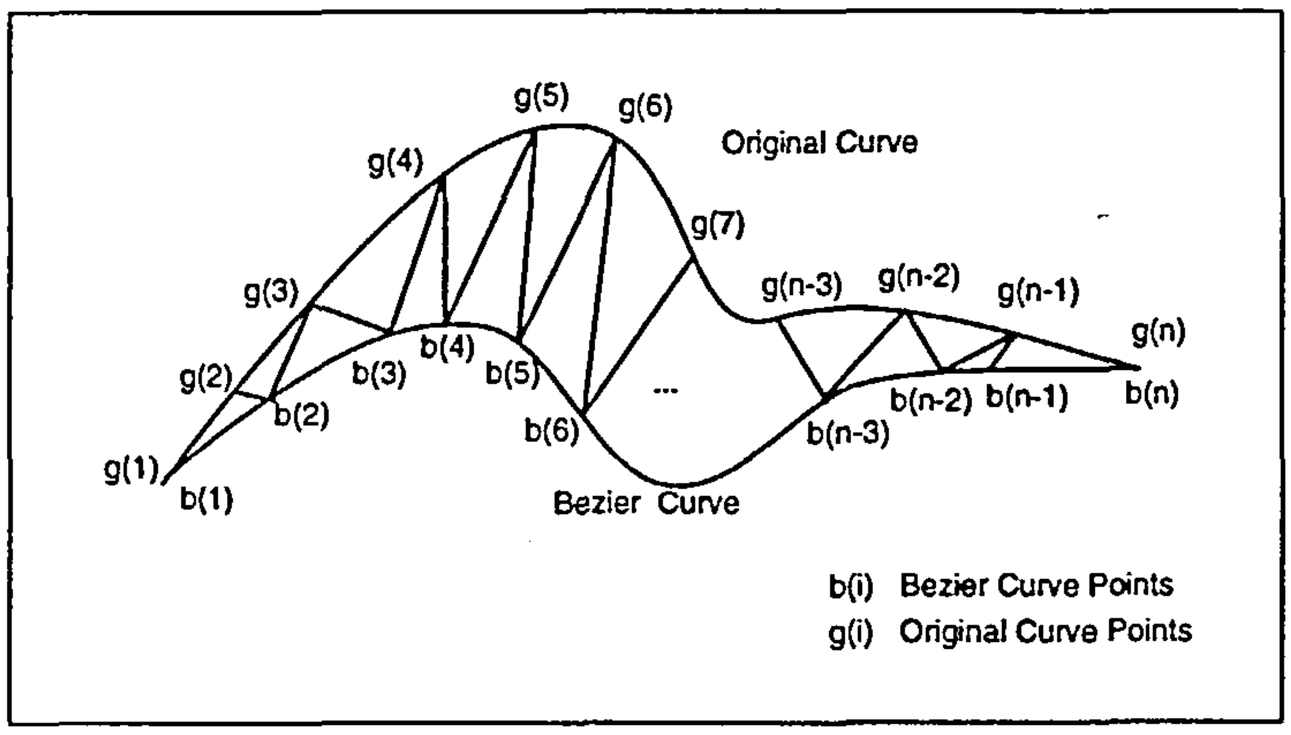

Figure C.2: Tension Surface Approximation Between Bezier Curve and Original Curve

curve) and by defining this surface as a *no-gravity, tension* surface, the best approximation to the curve would be given by minimizing the area of such a curve. The term *tension* surface denotes a surface that approximates the behaviour of a soap-film, i.e. exhibits surface tension. The *no-gravity* surface is a surface the shape of which is not influenced by gravity, i.e. the density per unit area is zero. In this simple case, given the two bounding curves, the tension surface is simply the surface created by connecting with straight lines the corresponding points of the two curves. If the exact correspondence between the points of the two curves was known, it would have been possible to minimize the sum of the squares of the distances between the corresponding points of the two curves. But since this is not available, the surface was approximated with a simple mesh of triangles, and the only requirement was that the Bezier curve be evaluated at the same number of points as the original curve. The geometry of the problem is shown in Fig. C.2.

If there are n points on each curve, then $2(n-2)$ triangles are formed and their sum approximates the area of the tension surface. Having as an objective the minimization of the surface area, the following constraints can be imposed:

- The coordinates of the first Bezier point must coincide with the coordi-

nates of the first point on the original curve.

- The coordinates of the last Bezier point must coincide with the coordinates of the last point on the original curve.

- The x-coordinates of the Bezier points have to be in ascending order.

- The slope of the Bezier curve at the first point must be the same with the slope of the original curve at the first point.

- The slope of the Bezier curve at the last point must be the same with the slope of the original curve at the last point.

To perform the optimization, the NPSOL optimizer was used successfully.

C.2.2 The Tensor Product Approach to the Bezier Surface Implementation

An intuitive definition of a surface is [53]: "A surface is the locus of a curve that is moving through space and thereby changing its shape". By formalizing this intuitive concept, a mathematical description of a surface can be obtained. The first assumption is that the moving curve is a Bezier curve of constant degree m. At any time, the moving curve is then determined by a set of control points. Each original control point moves through space on a curve. The next assumption is that this curve is also a Bezier curve, and that the curves on which the control points move are all of the same degree.

It can be easily shown that this definition leads to the Bezier patch definition in the previous sections.

C.3 Sample Implementation

The sample implementation consists of the design of the surface of the wing-fuselage junction fillet. As shown in Figure C.3, there are two surfaces that need to be defined, the upper and lower fillet surfaces. Each of them is bound by four curves, namely the fillet–fuselage interface line, the trailing edge, the fillet–wing interface line and the leading edge of the fillet. Since the tensor product surface design methodology will be used, the first step is to express those four curves in terms of their equivalent Bezier control points.

C.3.1 Implementation of Cubic Bezier Curves

To express the fillet–wing interface line in terms of a Bezier spline, it is sufficient to point out that the wing in some cases has a supercritical section which contains a *hump* on the lower surface. To accommodate such a hump, by implementing cubic Bezier curves, it will be necessary to use at least six piecewise cubic Bezier curves. The leading and trailing edge do not present any peculiarities in shape, and therefore could have been accommodated with two Bezier curves per spline, but to obtain more local control over the shape of the overall surface, three Bezier curves are used for the leading edge and the trailing edge spline.

Therefore, for an adequate representation of the fillet surface as shown in Figure C.3.1, a composite surface consisting of a six by three bicubic Bezier patches is used. This arrangement produces a total of twelve control points that are free to move while all the remaining from the patch net are constrained for the purpose of maintaining first derivative differentiability for the surface. Combined with the lower surface, a total of twenty two control points are available on the surface for shape design and optimization purposes.

In Figure C.3.1 the surface grid topology is shown. The wing–fillet interface

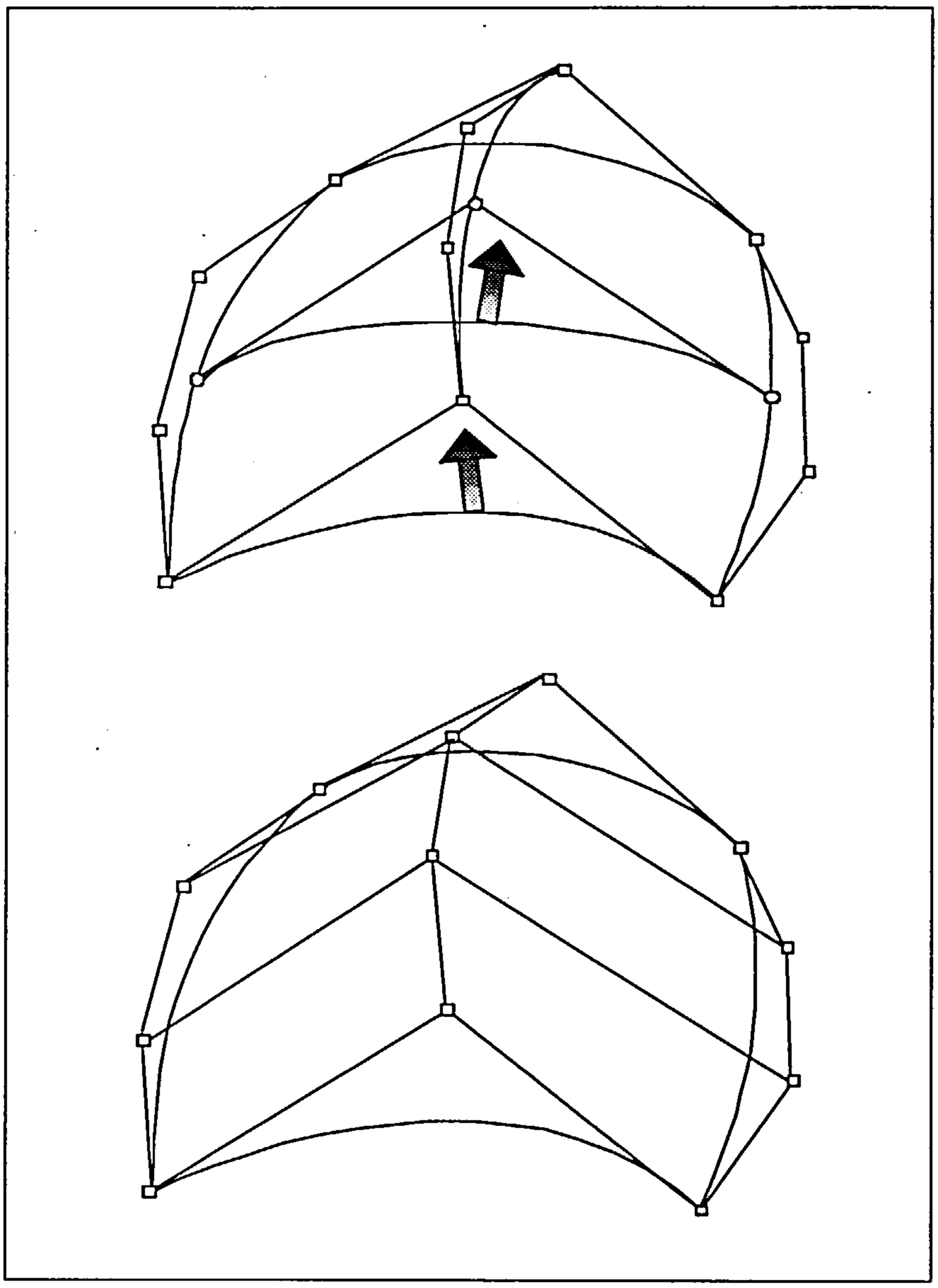

Figure C.3: Tensor Product Bezier Surfaces: Moving the control points of a curve along another Bezier curve (top). The final Bezier net (bottom).

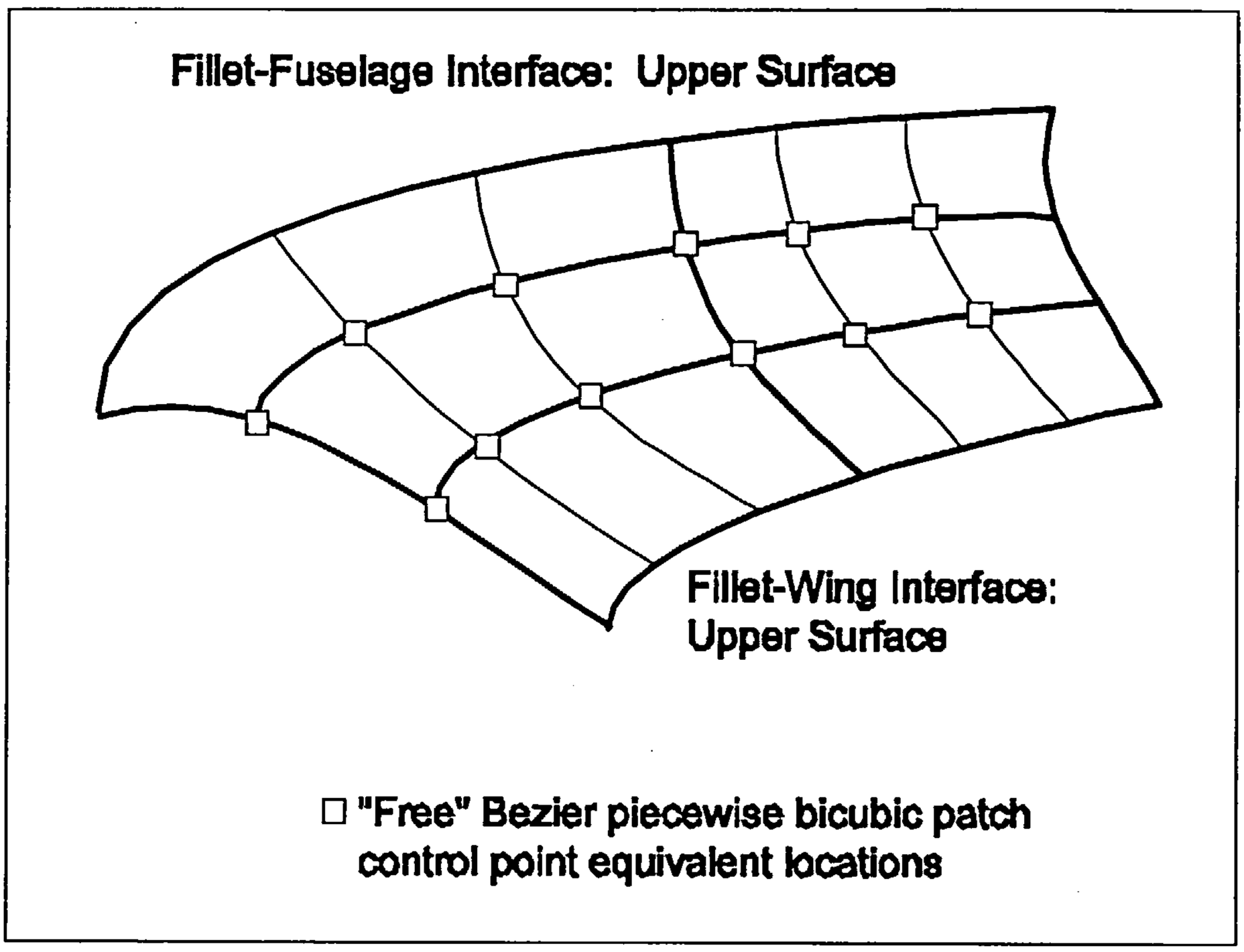

Figure C.4: Three by Six Bezier Bicubic Patch Distribution on the Upper Fillet Surface

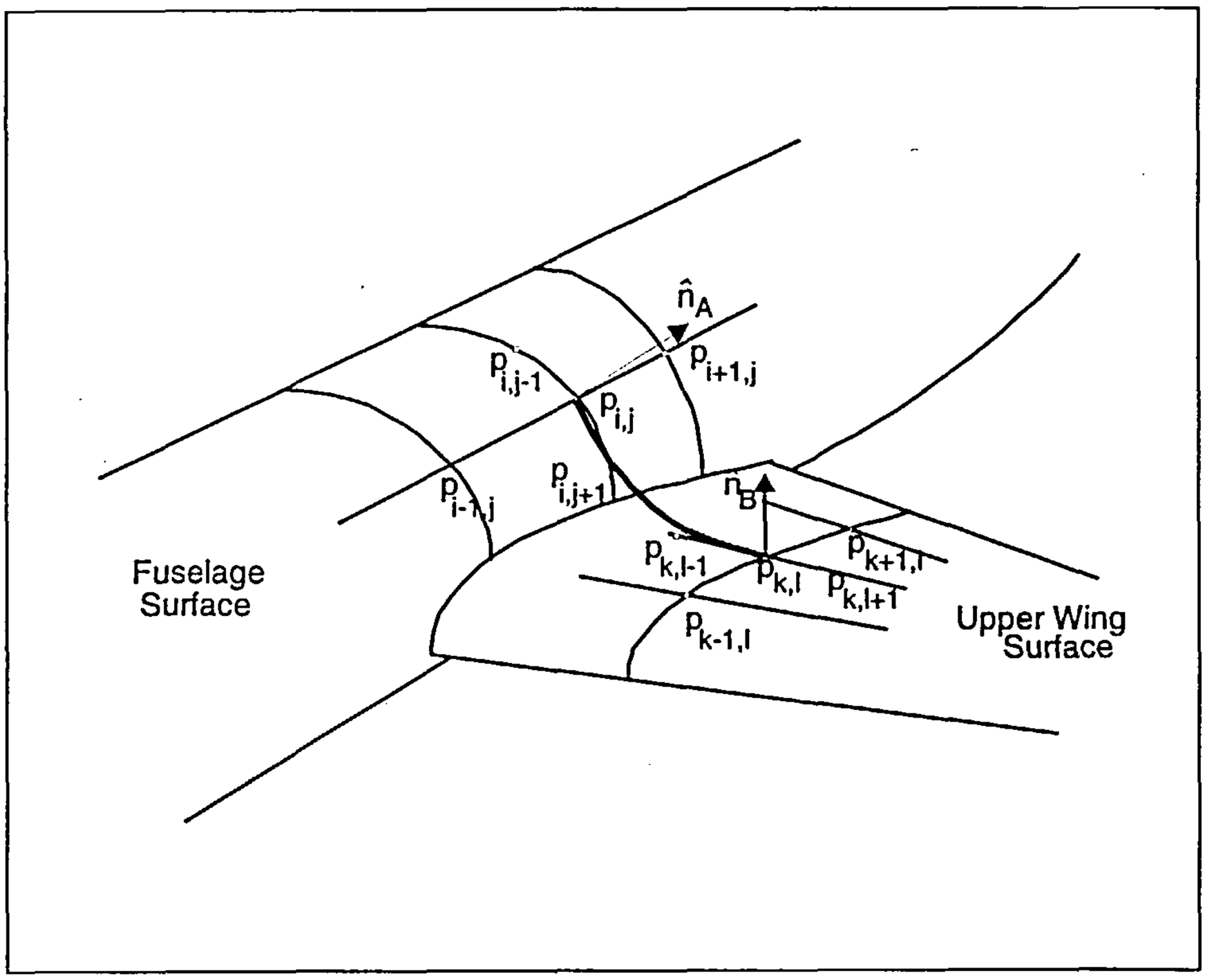

Figure C.5: Normal Vectors and Surface Grid Topology

line Bezier curves are produced by using the method outlined in Problem 1. Since the coordinates of the points on the wing are known, it is easy to approximate the derivative based on the coordinates of the points. For example, the derivative in the direction of the wing-fillet interface line of the wing surface at the point $\mathrm{p}_{i,j}$ is:

$$\vec{d}_p = \mathrm{p}_{k+1,1} - \mathrm{p}_{k-1,1}$$

and the unit directional vector is simply:

$$\hat{d}_p = \frac{\vec{d}_p}{|\vec{d}_p|}$$

The same approach yields the derivatives in the desired direction for the parameterization into a Bezier spline of the fillet-fuselage interface line.

To generate the 6×3 patch net with the tensor product approach, it was chosen to move the wing-fillet interface line towards the fuselage, and thus create the surface of the fillet. After defining the wing-fuselage and wing-fillet interface line in terms of Bezier control points, the leading edge and the trailing edge curves are defined. Once more the derivative vectors are available, and the curves are formed according to the methodology presented in Problem 1.

To complete the definition of the surface, the inner points must be determined. Since the wing-fillet interface line is moving towards the fuselage, this is the direction that the control points of the line must move, along the new Bezier curves. The derivatives in this direction, though, are not readily available. Nevertheless, it is possible to obtain the normal vectors at each point of interest. For example, the normal unit vector at the point $\mathbf{p}_{i,j}$ (point A in Figure C.3.1) is as follows:

$$\vec{n}_A = (\mathbf{p}_{i,j-1} - \mathbf{p}_{i,j+1}) \times (\mathbf{p}_{i-1,j} - \mathbf{p}_{i+1,j})$$

and

$$\hat{n}_A = \frac{\vec{n}_A}{|\vec{n}_A|}$$

Now that a normal unit vector $\hat{n}_A$ is available, the desired derivative unit vector must be obtained. Assuming a curve connecting point $\mathbf{p}_{i,j}$ (point A) to point $\mathbf{p}_{k,l}$ (point B) to find the unit derivative vector at A, point B is projected onto the plane defined by the normal $\hat{n}_A$ and point A. Then the desired derivative vector is:

$$\vec{d}_A = (\mathbf{p}_{k,l} - \mathbf{p}_{i,j}) - \left[\frac{(\mathbf{p}_{k,l} - \mathbf{p}_{i,j})}{|\mathbf{p}_{k,l} - \mathbf{p}_{i,j}|} \cdot \vec{n}_A\right] \hat{n}_A$$

and the unit vector then is:

$$\hat{n}_A = \frac{\vec{d}_A}{|\vec{d}_A|}$$

The same procedure is repeated for the determination of the derivative unit vector at point B.

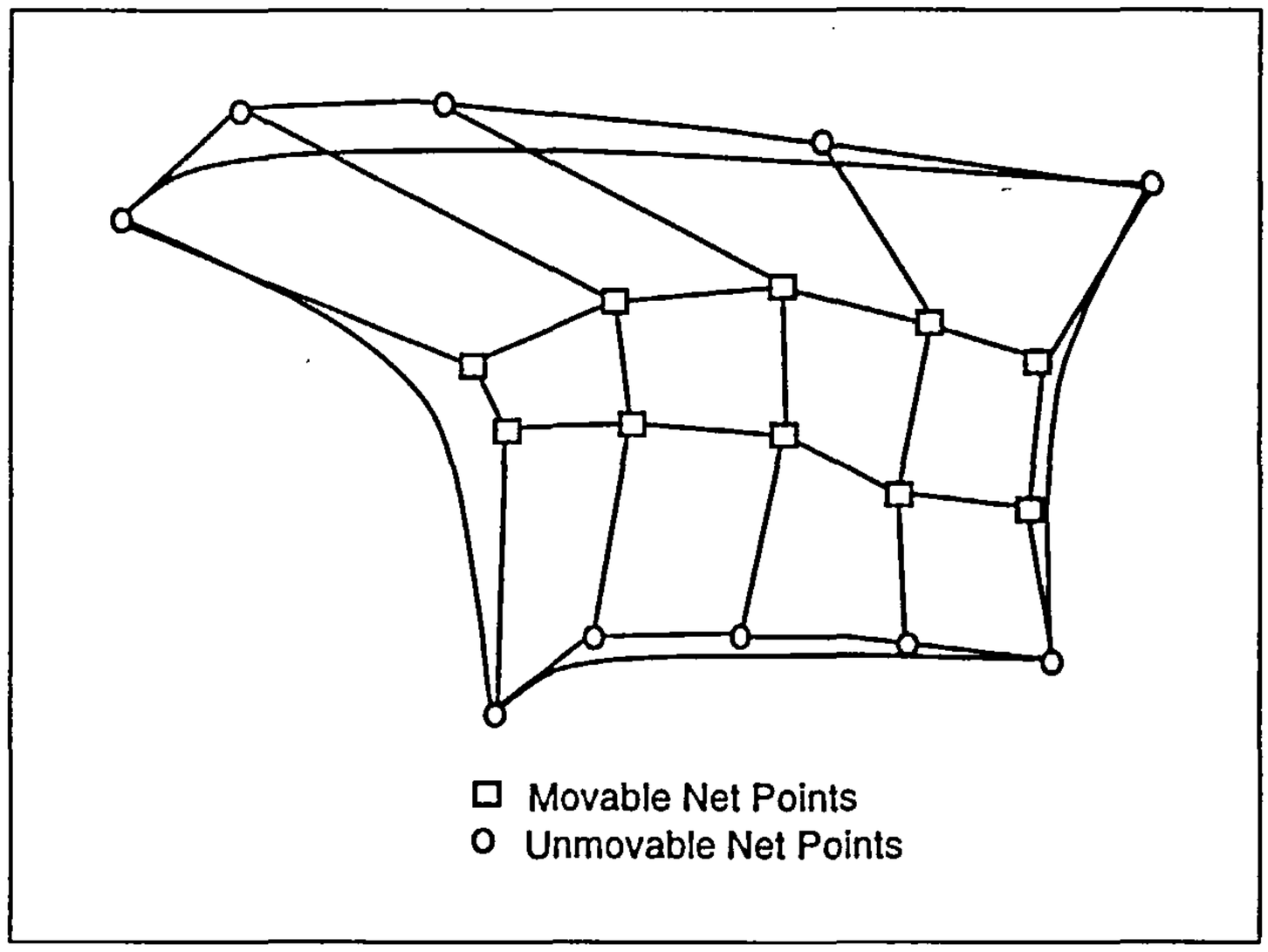

Figure C.6: Non-uniform 5x4 Point Bezier Surface Approximation of Fillet's Upper Wing Surface

Following this methodology, it is possible to obtain all 12 control points of the upper surface.

C.3.2 Implementation of the Higher Degree Bezier Curves and the Non-Uniform Bezier Surface Definition

To express the upper and lower wing surface in terms of a Bezier surface, two quartic Bezier curves approximated the fillet–wing and fillet–fuselage curves. The tensor surface was created by moving the first curve through space along five cubic Bezier curves controlling the shape of the surface. C.3.2 The surface control net was set up according to the following requirements:

- The slope in the spanwise direction at the Bezier points representing the fillet-wing intersection curve must be the same as the actual wing surface slope in the spanwise direction at the points with corresponding x and y coordinates.

- The slope in the circumferential direction at the three inner Bezier points representing the fillet-fuselage intersection curve must be the same as the actual fuselage surface slope in the circumferential direction at the points with corresponding x-coordinates.

- The slope along the length of the fuselage at the two outer Bezier points representing the fillet-fuselage intersection curve must be the same as the actual fuselage surface slope along the length of the fuselage at the points with corresponding x-coordinates.

- The slope at the first point of the Bezier curve describing the leading edge of the fillet must be the same as the slope along the length of the fuselage of the first point on the fillet-fuselage intersection curve.

- The slope at the last point of the Bezier curve describing the leading edge of the fillet must be the same as the slope in the spanwise direction of the first point on the fillet-wing intersection curve.

- The slope at the first point of the Bezier curve describing the trailing edge of the fillet must be the same as the slope along the length of the fuselage of the last point on the fillet-fuselage intersection curve.

- The slope at the last point of the Bezier curve describing the trailing edge of the fillet must be the same as the slope in the spanwise direction of the last point on the fillet-wing intersection curve.

All of the above are direct consequences of the definition of the tensor product surface. These slope requirements ensure that the the transition from the fuselage surface to the fillet surface and from the fillet surface to the wing surface, will be described by an at least once differentiable surface, eliminating any corners and producing a smooth transition.

As a direct result of the above restrictions, and considering the fillet-wing and fillet-fuselage bounding curves and their corresponding Bezier points as fixed, there is a total of ten Bezier Net points that can be moved (See Fig. C.3.2. These ten points can only be moved along given directions, along given line paths, namely the slope lines, a direct consequence of the above requirements imposed on the slope of the surface at the bounding curves.

Appendix D

The Navier Stokes Solver

D.1 Introduction

In the thin-layer approximation to the Navier–Stokes equations, the viscous terms containing derivatives in the directions parallel to the body surface are neglected since they are many times substantially smaller than viscous terms containing derivatives normal to the wall, while the remaining terms in the momentum equations are retained. One of the principal advantages of retaining the terms which are normally neglected in boundary–layer theory is that separated and reverse flow regions can be computed in a straightforward manner. Also, flows containing a large normal pressure gradient can be readily computed.

The thin–layer approximation concept also arises from a detailed examination of typical high Reynolds number computations involving the complete Navier–Stokes equations[8]. In these computations a substantial fraction of the available computer storage and time is expended in resolving the normal gradients in the boundary layer since a highly stretched grid is required. As a result, the gradients parallel to the body surface are usually not resolved in an adequate manner even though the corresponding viscous terms are re-

tained in the computations. Hence, for many Navier–Stokes computations it makes sense to drop those terms that are not being adequately resolved provided that they are reasonably small. This leads to the use of the thin–layer Navier–Stokes equations.

D.2 The EAGLE TLNS Solver

The latest module of the EAGLE Flow Solver program is a multi-block implicit steady-state thin-layer Navier-Stokes solver. It is specifically designed to interface with the numerical grid generation system, to solve for the freestream aerodynamic characteristics of airframe configurations, as well as the interference flowfield associated with complex configurations. Program EAGLE–Flow Solver has been a joint development effort between the Air Force Armament Laboratory's (AFATL) Aerodynamics Branch and Mississippi State University's Department of Aerospace Engineering. Unfortunately neither a User's Manual nor a Reference Manual was available for the Navier-Stokes solver. In an attempt to facilitate the future use of such a powerful tool, a description of the required inputs for the NAMELIST entries required is provided.

D.3 EAGLE Grid Code Input Listing

A typical input data file is shown below.

```
$ FINPUT CFL=7.0, FSMACH=0.85, LIFTAX=+3, JFREQ=40,
   PRINT=123, NPR=1, NIT=2000, NCYC=1, NSURF=4,SPLIT=2,
   LIMIT=2,RESTRT=0,
   NZPG=75, NGRAD=25,NBLK=2, RORDER=2 $
$ VINPUT INVISC=0,LTSTEP=100,MTSTEP=100 $
$ ROTATE TRANS = 0.00, PHIY = 4.00 $
Body With Wing
$ SURFACE SREF= 1, XREF=6.238  $
Upper Wing
```

```
$ SURFACE SREF=2, XREF=6.238 $
Lower Wing
$ SURFACE SREF=3, XREF=6.238 $
Body Alone
$ SURFACE SREF=4, XREF=6.238 $
$ BCIN BCTYPE=1, SURF=1,4,BLKA=1,STARTA=1,1,1, ENDA=50,16,1$
$ BCIN BCTYPE=1, SURF=1,4,BLKA=2,STARTA=1,1,1, ENDA=50,5,1$
*
*  The Wing and the Fillet
*
 $ BCIN BCTYPE=1, SURF=1,2,BLKA=1,STARTA=15,16,1, ENDA=34,16,25 $
 $ BCIN BCTYPE=1, SURF=1,3,BLKA=2,STARTA=15,5,1, ENDA=34,5,25$
 $ BCIN BCTYPE=2, BLKA=1,STARTA=1,1,35, ENDA=50,16,35 $
 $ BCIN BCTYPE=2, BLKA=2,STARTA=1,1,35, ENDA=50,5,35 $
 $ BCIN BCTYPE=3, BLKA=1,STARTA=1,1,1, ENDA=50,1,35 $
 $ BCIN BCTYPE=3, BLKA=2, STARTA=1,1,1, ENDA=50,1,35 $
*
*  The Remaining Surface 4
*
 $ BCIN BCTYPE=4, BLKA=1,STARTA=1,16,1, ENDA=14,16,35,
                          BLKB=2,STARTB=1,5,1, ENDB=14,5,35 $
 $ BCIN BCTYPE=4, BLKA=1,STARTA=35,16,1, ENDA=50,16,35,
                          BLKB=2,STARTB=35,5,1,ENDB=50,5,35 $
 $ BCIN BCTYPE=4, BLKA=1,STARTA=15,16,26, ENDA=34,16,35,
                          BLKB=2,STARTB=15,5,26, ENDB=34,5,35 $
* $ BCIN BCTYPE=4, BLKA=1,STARTA=1,16,1, ENDA=50,16,35,
                          BLKB=2, STARTB=1,5,1,ENDB=50,5,35 $
 $ BCIN BCTYPE=6, BLKA=1,STARTA=1,1,1, ENDA=1,16,35 $
 $ BCIN BCTYPE=6, BLKA=2,STARTA=1,1,1, ENDA=1,5,35 $
 $ BCIN BCTYPE=6, BLKA=1,STARTA=50,1,1, ENDA=50,16,35 $
 $ BCIN BCTYPE=6, BLKA=2,STARTA=50,1,1,ENDA=50,5,35 $
 $ BCIN BCTYPE=0 $
 $ TMODEL MTYPE=2,BLKA=1,STARTA=1,1,1,ENDA=50,16,35 $
 $ TMODEL MTYPE=2,BLKA=2,STARTA=1,1,1,ENDA=50,5,35 $
* $ TMODEL MTYPE=1,BLKA=1,STARTA=1,1,1,ENDA=36,16,5 $
* $ TMODEL MTYPE=1,BLKA=2,STARTA=1,1,1,ENDA=36,5,5 $
* $ TMODEL MTYPE=1,BLKA=1,STARTA=10,10,1,ENDA=36,16,10 $
* $ TMODEL MTYPE=1,BLKA=2,STARTA=10,1,1,ENDA=36,5,10 $
 $ TMODEL MTYPE=0 $
*
*
*
```

```
$ END $
```

D.4 EAGLE Grid Code Output

The output from the data file described above is shown in the following pages.

```
*************************************************
*                                               *
*          program misst- flow solver           *
* multi-block implicit steady-state tlns algorithm *
*          misst.f 1.1 10/5/90                  *
*          cray ymp version (25 sept 1990)      *
*                                               *
*************************************************

     air force armament laboratory (afatl)
     aeromechanics division (afatl/fx)
     aerodynamics branch (afatl/fxa)
     computational fluid dynamics section (cfd)
     eglin afb, fl 32542-5242

 ************************************************

restart
grid.dat.4
 the grid file is  grid.dat.4
 the restart file is  restart

 ******************** echo check ***********************

   namelist /finput/ :
courant number (cfl) = 7.
freestream mach number (fsmach) = 0.8500000000000014
frequency of flux jacobian matrix updates
 (jfreq) = 40
number of zero pressure gradient bc iterations
 (nzpg) = 75
```

```
number of gradually applied zpg bc iterations
 (ngrad) = 25
lift will be calculated for the #3 axis
 (liftax)
total number of surfaces (nsurf) = 4
total number of iterations (npr)*(nit) = 2000

 namelist /vinput/ :

 number of viscous iterations = 2000
 local time-step => 100
  number of local time-steps = 9999
 minimum time-step => 100
  number of minimum time-steps = 0
  minimum time-step size (dtmin) = 0.1000000000000001


**********************************************************

 computational region (grid):

  block      ni     nj     nk
     1       50       16       35
     2       50        5       35

**********************************************************

 namelist /rotate/ :

 rotation angle sequence (trans) = 0,  0,  0
 rotation angles :
   phix = 0.
   phiy = 0.
   phiz = 0.

**********************************************************

 splitting technique :

   roe averaged (flux-difference)
    split = 2
    limit = 2
```

```
     rorder= 2

*****************************************************

  namelist /surface/ :

  surface reference number (sref) : 1

      Body With Wing

      reference area          (aref) = 1.
      reference length        (lref) = 1.
      "x" reference           (xref) = 6.238
      "y" reference           (yref) = 0.
      "z" reference           (zref) = 0.

  ...

*****************************************************
calculating max dimensions in x,y,z
set memory for bc check planes
 initialize all blocks

  total in-core memory allocated : 8935466

0                      1 printouts every   400 cycles
step ib rtmax irm jrm krm rtrms    etmax       etrms     l2norm  nsup
0 1 -2.637E-03 40 15 2  0.000E+00 -7.587E-03  0.000E+00 -8.413E-02 0
0 2 -3.090E-03  4 3  2 -2.261E-01 -7.951E-03 -2.642E-01 -8.128E-02 0

1 1 -5.500E-03 40 15 2  3.147E-01 -1.602E-02  3.161E-01  2.317E-01 0
1 2 -5.583E-03  4  3 2  6.811E-02 -1.461E-02  3.481E-02  2.315E-01 0

2 1  8.354E-03 10 14 2  4.990E-01 -2.400E-02  5.001E-01  4.151E-01 0
2 2  7.569E-03 40  4 2  2.364E-01 -1.987E-02  2.071E-01  4.207E-01 0

<<  Steps up to last iteration omitted >>

     istep = 400    block = 1
```

```
  2                    3                    4
     x        cp          x        cp          x        cp
2 -15.9430 1.3919    -15.9430 1.3499    -15.9430 1.3301
3 -15.6415 1.4586    -15.6415 1.5657    -15.6415 1.5693
4 -14.8925 0.7188    -14.9082 0.7698    -14.9075 0.7250

...

49 54.3070 -0.3599    54.3070 -0.3213    54.3070 -0.3167
50 55.8070 -0.1737    55.8070 -0.1452    55.8070 -0.1595
 5                   6                   7
      x        cp          x        cp          x        cp
2 -15.9430 1.2942    -15.9430 1.2572    -15.9430 1.2119
3 -15.6415 1.5806    -15.6415 1.5968    -15.6415 1.6028

...

22 21.5127 -0.0841    21.8835 -0.0822    22.2535 -0.0818
23 22.8527 -0.0862    23.1772 -0.0839    23.5012 -0.0829
24 24.3095 -0.0884    24.5840 -0.0862    24.8580 -0.0853
25 25.9460 -0.0919    26.1640 -0.0897    26.3818 -0.0886
  34
     x        cp
   2     20.2217    -0.1687
   3     20.0675    -0.1156
   4     19.8527    -0.1065
   5     19.5722    -0.0970
   6     19.2470    -0.0916
   7     18.9027    -0.0870

...

  24     25.1315    -0.0869
  25     26.5992    -0.0904

   istep = 400   block = 2
 2                    3                    4
      x        cp          x        cp          x        cp
2 -15.9430 -0.7709    -15.9430 -0.7688    -15.9430 -0.7766
3 -15.6415 -0.7193    -15.6415 -0.7565    -15.6415 -0.7963

...
```

```
48      51.2780      0.0289
49      54.3070      0.4890
50      55.8070      0.2711


  istep = 400   block = 2
16                  17                  18
     x       cp          x       cp          x       cp
2 0.1303 -0.2009     1.1081 -0.2295     2.1799 -0.2354
3 0.3701 -0.2242     1.3805 -0.2361     2.4375 -0.2377

...

  23      23.8227    -0.1199
  24      25.1300    -0.1191
  25      26.5985    -0.1178

reference
surface   name          area          x             y             z
1   Body With Wing      1.00000     6.23800     0.00000     0.00000

2   Upper Wing          1.00000     6.23800     0.00000     0.00000

3   Lower Wing          1.00000     6.23800     0.00000     0.00000

4   Body Alone          1.00000     6.23800     0.00000     0.00000


     pressure
surface   name              forces
       1   Body With Wing   fxp=        8.881952
       2   Upper Wing       fxp=       -1.282431
       3   Lower Wing       fxp=        0.985703
       4   Body Alone       fxp=        9.178680

surface   name              forces                          moments
1   Body With Wing   fx =        88.881952    mx =      1292.129459
     fy =         41.469404    my =      -650.863307
     fz =        121.003551    mz =       586.575456
2   Upper Wing         fx =      11.282431    mx =       629.158062
     fy =          0.295376    my =      -321.403172
```

```
      fz =        40.324046    mz =         4.683396
3   Lower Wing        fx =        15.985703    mx =       703.404763
      fy =        -7.229705    my =      -399.839426
      fz =        53.559420    mz =       -50.994865
4   Body Alone        fx =        59.178680    mx =       -40.433366
      fy =        48.403733    my =        70.379291
      fz =        27.120086    mz =       632.886925

coefficients
surface   name              forces                        moments
1   Body With Wing    cx =        88.881952    cmx =      1292.129459
      cy =        41.469404    cmy =      -650.863307
      cz =       121.003551    cmz =       586.575456
2   Upper Wing        cx =        11.282431    cmx =       629.158062
      cy =         0.295376    cmy =      -321.403172
      cz =        40.324046    cmz =         4.683396
3   Lower Wing        cx =        15.985703    cmx =       703.404763
      cy =        -7.229705    cmy =      -399.839426
      cz =        53.559420    cmz =       -50.994865
4   Body Alone        cx =        59.178680    cmx =       -40.433366
      cy =        48.403733    cmy =        70.379291
      cz =        27.120086    cmz =       632.886925

    surface   name              forces                    coefficients
1   Body With Wing    fl =       121.003551    cl  =       121.003551
      fd =        88.881952    cd  =         8.881952
      fs =        41.469404    cs  =        41.469404
2   Upper Wing        fl =        40.324046    cl  =        40.324046
      fd =        11.282431    cd  =        -1.282431
      fs =         0.295376    cs  =         0.295376
3   Lower Wing        fl =        53.559420    cl  =        53.559420
      fd =        15.985703    cd  =         0.985703
      fs =        -7.229705    cs  =        -7.229705
4   Body Alone        fl =        27.120086    cl  =        27.120086
      fd =        59.178680    cd  =         9.178680
      fs =        48.403733    cs  =        48.403733

  total cpu time used: 15694.49665961007
logout
```

Appendix E

The NPSOL Optimization Package

The NPSOL package is designed to minimize smooth functions subject to constraints, which may include simple bounds, linear constraints, and smooth nonlinear constraints. The software uses a sequential quadratic programming algorithm, where bounds, linear constraints and nonlinear constraints are treated separately. Unlike other software for optimization (e.g. MINOS), NPSOL stores all matrices in dense format, and is therefore not intended for large sparse problems. NPSOL is available from the Office of Technology Licensing at Stanford University [65].

E.1 Example NPSOL Application

The use of the package will be demonstrated here on a simple example of minimizing a nonlinear objective function with linear constraints.

Assuming an objective function of the form

$$f(x) = -x_1 x_2 x_3$$

the problem is stated as:

Minimize $f(x)$ subject to the following constraints:

$$x_1 \leq 42$$

$$x_1 + 2x_2 + 2x_3 \leq 72$$

for positive x_1, x_2, x_3.

Since NPSOL solves nonlinear programming problems of the form:

$$Minimize \quad F(x)$$

$$subject \quad to \quad LB < \left\{ \begin{array}{c} x \\ Ax \\ c(x) \end{array} \right\} < UB$$

where

$F(x)$ is a smooth scalar function

A is a constant matrix

$c(x)$ is a vector of smooth nonlinear functions

LB is the lower bound for the constraints and

UB is the upper bound for the constraints.

the simple problem above will have to be brought in this form.

In this case, there are no nonlinear terms, and therefore the matrix $c(x)$ does not exist. Matrix A is the linear constraint coefficient matrix, which in this example's case is the following 2×3 matrix:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 2 & 2 \end{bmatrix}$$

so that $LB \leq Ax \leq UB$ where x is the column vector containing the three variables:

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

LB is the column matrix formed by the lower bounds of the variables and the constraints in this order:

$$LB = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

and similarly UB is the column matrix formed by the upper bounds of the variables and the constraints in this order:

$$UB = \begin{bmatrix} +\infty \\ +\infty \\ +\infty \\ 42.0 \\ 72.0 \end{bmatrix}$$

A number of options may be read through an input file, but in this case, no options are required aside from the defaults. For clarity, the call invoking the reading of the options file is left within the code presented below.

The objective function must be defined within a subroutine, as well as its first derivatives. The partial derivatives of the objective function with respect to each of the variables are stored in a column vector, where the first element is the partial derivative of the objective function with respect to the first variable, the second element is the partial derivative of the objective function with respect to the second variable and so on. This is accomplished in the code below through the subroutine `OBJFN1`.

If any nonlinear constraints were present, they would have to be defined in the subroutine CONFN1. Even though this case contains no nonlinear constraints, the format of a typical such subroutine is maintained for completeness.

Finally, the problem is solved by invoking the NPSOL routine and the results are printed out.

E.2 NPSOL Problem Description Listing

The invoking program for the simple function of three variables subject to the two constraints is shown below along with the definition of the objective function and the non-functional nonlinear constraint subroutine. The template for the routine is freely available from the distribution. Using this template, one can define the bunction described above, along with all the constraints as shown below:

```
program dpnpmain
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*     FILE NPMAIN FORTRAN
*
*     Sample program for NPSOL Version 4.04  August 1986.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**********************************************************
*     OBJECTIVE OF THREE VARIABLES WITH NON-LINEAR  AND
*     LINEAR CONTSTRAINTS.  (VOLUME CONSTRAINT)
*
*     IMPLEMENTED BY A.H. HADJIILIAS
*     OCTOBER 1995
*     CRANFIELD UNIVERSITY
*     COLLEGE OF AERONAUTICS
**********************************************************
      IMPLICIT           DOUBLE PRECISION(A-H,O-Z)

*     Set the declared array dimensions.
*     NROWA  = the declared row dimension of  A.
```

```
*     NROWJ  = the declared row dimension of  CJAC.
*     NROWR  = the declared row dimension of  R.
*     MAXN   = max no. of variables allowed for.
*     MAXBND = max no. of variables+lin.&nonlin. constrnts.
*     LIWORK = the length of the integer work array.
*     LWORK  = the length of the double precision work array.

      PARAMETER          (NROWA  =  1, NROWJ  =  1, NROWR =   10,
     $                    MAXN   =  1, LIWORK =  10, LWORK = 50,
     $                    MAXBND =  MAXN + NROWA + NROWJ)

      INTEGER            ISTATE(MAXBND)
      INTEGER            IWORK(LIWORK)
      DOUBLE PRECISION   A(NROWA,MAXN)
      DOUBLE PRECISION   BL(MAXBND), BU(MAXBND)
      DOUBLE PRECISION   C(NROWJ), CJAC(NROWJ,MAXN), CLAMDA(MAXBND)
      DOUBLE PRECISION   OBJGRD(MAXN), R(NROWR,MAXN), X(MAXN)
      DOUBLE PRECISION   WORK(LWORK)
      EXTERNAL           OBJFN1,CONFN1,OBJFN2, CONFN2

      PARAMETER          (ZERO = 0.0, ONE = 1.0)

*     Set the actual problem dimensions.
*     N      = the number of variables.
*     NCLIN  = the number of general linear constraints (may be 0).
*     NCNLN  = the number of nonlinear constraints (may be 0).

      N      = 3
      NCLIN  = 2
      NCNLN  = 0
      NBND   = N + NCLIN + NCNLN

*     ------------------------------------------------
*     Assign file numbers and the data arrays.
*     NOUT   = the unit number for printing.
*     IOPTNS = the unit number for reading the options file.
*     Bounds  .ge.    BIGBND  will be treated as plus  infinity.
*     Bounds  .le.  - BIGBND  will be treated as minus infinity.
*     A      = the linear constraint matrix.
*     BL     = the lower bounds on  x,  a'x  and  c(x).
*     BU     = the upper bounds on  x,  a'x  and  c(x).
*     X      = the initial estimate of the solution.
```

```
*     ------------------------------------------------------
      NOUT   = 6
      IOPTNS = 5
      BIGBND = 1.0D+15

*     Set the matrix  A.

      DO 40 J = 1, N
 DO 30 I = 1, NCLIN
    A(I,J) = ZERO
   30    CONTINUE
   40 CONTINUE
      A(1,1) = ONE
      A(2,2) = 2.0D0
      A(2,3) = 2.0D0
      A(2,1) =  ONE

*     Set the bounds.

      DO 50 J  =  1, NBND
 BL(J) = -BIGBND
 BU(J) =  BIGBND
   50 CONTINUE
      BL(1)  =  0.0D0
      BL(2)  =  0.0D0
      BL(3)  =  0.0D0
      BL(4)  =  ZERO
      BL(5)  =  ZERO

      BU(1)  =  BIGBND
      BU(2)  =  BIGBND
      BU(3)  =  BIGBND
      BU(4)  =  42.0D0
      BU(5)  =  72.0D0

*     Set the initial estimate of  X.

      X(1)   =  .1
      X(2)   =  .125
      X(3)   =  .666666
```

```
*     ------------------------------------------------------
*     Read the options file.
*     ------------------------------------------------------

      CALL NPFILE( IOPTNS, INFORM )
      IF (INFORM .NE. 0) THEN
      WRITE (NOUT, 3000) INFORM
      STOP
      END IF

*     ------------------------------------------------------
*     Solve the problem.
*     ------------------------------------------------------

      CALL NPSOL ( N, NCLIN, NCNLN, NROWA, NROWJ, NROWR,
     $             A, BL, BU,
     $             CONFN1, OBJFN1,
     $             INFORM, ITER, ISTATE,
     $             C, CJAC, CLAMDA, OBJF, OBJGRD, R, X,
     $             IWORK, LIWORK, WORK, LWORK )
      IF (INFORM .GT. 0) GO TO 900

*     -----------
*     Error exit.
*     -----------

  900 WRITE (NOUT, 3010) INFORM
      STOP

 3000 FORMAT(/ ' NPFILE terminated with  INFORM =', I3)
 3010 FORMAT(/ ' NPSOL  terminated with  INFORM =', I3)

*     End of the example program for NPSOL.

      END
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      SUBROUTINE OBJFN1( MODE, N, X, OBJF, OBJGRD, NSTATE )
      IMPLICIT           DOUBLE PRECISION(A-H,O-Z)
      DOUBLE PRECISION   X(N), OBJGRD(N)
```

```
*-----------------------------------------------------------
*     OBJFN1  computes the value and first derivatives
*      of the nonlinear objective function.
*-----------------------------------------------------------
      OBJF = -X(1)*X(2)*X(3)
      OBJGRD(1) =   -X(2)*X(3)
      OBJGRD(2) =   -X(1)*X(3)
      OBJGRD(3) =   -X(1)*X(2)
      RETURN

*     End of  OBJFN1.

      END
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      SUBROUTINE CONFN1( MODE, NCNLN, N, NROWJ,
     $                   NEEDC, X, C, CJAC, NSTATE )

      IMPLICIT           DOUBLE PRECISION(A-H,O-Z)
      INTEGER            NEEDC(*)
      DOUBLE PRECISION   X(N), C(*), CJAC(NROWJ,*)

*-----------------------------------------------------------
*     CONFN1  computes the values and first derivatives
*     of the nonlinear  constraints.
*
*     The zero elements of Jacobian matrix are set only once.
*     This occurs during the first call to CONFN1  (NSTATE = 1).
*-----------------------------------------------------------
      PARAMETER          (ZERO = 0.0, TWO = 2.0)

      IF (NSTATE .EQ. 1) THEN
      END IF

      IF (NEEDC(1) .GT. 0) THEN
C(1)        =   X(1)**2
CJAC(1,1)   =   TWO*X(1)
      END IF
```

```
      RETURN

*     End of  CONFN1.

      END
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

E.3 NPSOL Output File

The NPSOL output file is shown in the following pages. The output produces initially a listing of the options used, and then a table containing the general description and requirements of the problem along with the type of information that is available to or required from the solver.

A detailed convergence history is then provided providing among other information the objective function values along the optimization process for every iteration. A summary including the number of functional evaluations, iterations and derivative evaluations is also provided.

Finally, the converged values for the variables are presented along with the status of the constraints at the converged value. The output is completed with the listing of the objective value at the converged optimum.

```
OPTIONS file
------------

     begin
     end
```

```
NPSOL  ---  Version 4.05   Nov  1989
==================================
```

```
Parameters
----------

Linear constraints.....        2 Linear feasibility.....  1.05E-08
COLD start.............
Variables..............        3 Infinite bound size....  1.00E+20
Crash tolerance........  1.00E-02
Step limit.............  2.00E+00 Infinite step size.....  1.00E+20

Nonlinear constraints..        0 Optimality tolerance...  3.26E-12
Function precision.....  4.37E-15
Nonlinear Jacobian vars        3 Nonlinear feasibility..  1.05E-08
Nonlinear objectiv vars        3 Linesearch tolerance...  9.00E-01
EPS (machine precision)  1.11E-16 Derivative level.......        3
Verify level...........        0

Major iterations limit.       50 Major print level......       10
Minor iterations limit.       50 Minor print level......        0
RUN loaded from file...        0 RUN to be saved on file        0
Save frequency.........       51

Workspace provided is     IW(      11),  W(     100).
To solve problem we need  IW(      11),  W(     100).


Verification of the objective gradients.
----------------------------------------

The objective gradients seem to be ok.

Directional derivative of the objective   -6.86666667E+01
Difference approximation                  -6.86666675E+01


 Itn ItQP     Step Nfun     Objective Bnd Lin  Nz  Norm Gf
   0    3 0.0E+00    1 -8.000000E+02   1   1   1  2.2E+02
   1    1 3.3E-01    3 -1.528418E+03   1   1   1  2.2E+02
   2    2 4.1E-01    5 -2.339477E+03   0   1   2  3.5E+02
   3    1 1.0E+00    6 -2.956131E+03   0   1   2  4.8E+02
   4    1 1.0E+00    7 -3.437287E+03   0   1   2  4.3E+02
   5    1 1.0E+00    8 -3.455396E+03   0   1   2  4.3E+02
```

```
   6     1  1.0E+00      9 -3.455998E+03    0    1    2  4.3E+02
   7     1  1.0E+00     10 -3.456000E+03    0    1    2  4.3E+02
   8     1  1.0E+00     11 -3.456000E+03    0    1    2  4.3E+02
   9     1  1.0E+00     12 -3.456000E+03    0    1    2  4.3E+02
  10     1  1.0E+00     13 -3.456000E+03    0    1    2  4.3E+02
Itn      Norm Gz  Cond H Cond Hz  Cond T Conv
  0     1.8E+01  1.E+00  1.E+00  2.E+00 F FF
  1     1.2E+02  2.E+01  1.E+00  2.E+00 F FF
  2     1.1E+02  2.E+01  2.E+00  3.E+00 F F
  3     1.6E+02  4.E+01  2.E+00  3.E+00 F FT
  4     2.7E+01  4.E+01  2.E+00  3.E+00 F F
  5     4.7E+00  4.E+01  2.E+00  3.E+00 F FT
  6     2.9E-01  4.E+01  2.E+00  3.E+00 F FT
  7     2.5E-02  3.E+01  2.E+00  3.E+00 F FT
  8     2.3E-03  3.E+01  2.E+00  3.E+00 F TT
  9     8.5E-05  3.E+01  2.E+00  3.E+00 F TT
 10     2.4E-06  3.E+01  2.E+00  3.E+00 T TT

Exit  NP phase.  INFORM = 0 MAJITS = 10   NFUN = 13   NGRAD = 13

Variable          State      Value          Lower bound
VARBL  1            FR     24.00000       0.0000000E+00
VARBL  2            FR     12.00000       0.0000000E+00
VARBL  3            FR     12.00000       0.0000000E+00

Upper bound  Lagr multiplier     Residual
0.1000000E+16   0.0000000E+00       24.00
0.1000000E+16   0.0000000E+00       12.00
0.1000000E+16   0.0000000E+00       12.00

Linear constr   State       Value          Lower bound
LNCON  1          FR     24.00000       0.0000000E+00
LNCON  2          UL     72.00000       0.0000000E+00

Upper bound  Lagr multiplier     Residual
42.00000        0.0000000E+00       18.00
72.00000        -144.0000        -0.1421E-13

Exit NPSOL - Optimal solution found.

Final nonlinear objective value =    -3456.000
```

```
NPSOL  terminated with  INFORM =  0
```

Appendix F

Program Listings

The programs that were either created or modified for the simulations performed in this thesis can be classified into two categories: the programs written for analysis purposes and the programs written for optimization purposes. A description of these two categories follows.

F.1 The Analysis Programs

The analysis part of the thesis consists of the following elements:

1. Fillet surface definition according to input values[1]

2. Constraint check on generated fillet surface and volume

3. Incorporation of surface into the geometry definition of the aircraft

4. Grid Creation

5. Navier-Stokes Solution

[1]This first action performs the approximation of the fillet bounding curves by quartic Bezier curves, in which the optimizer NPSOL is used (Appendix E).

Each of the above represents one or more programming modules that perform a specific function, as shown in Figure F.1. A breakdown of each block along with the complete listing of the programs written and used is presented below. The language the programs were written in is FORTRAN 77. Some important text file processing was performed using GAWK, a special-purpose programming language that makes it possible to handle simple data-reformatting jobs easily with just a few lines of code. GAWK is the GNU implementation of AWK, and is upward compatible with both the System V Release 4 version of AWK and with the POSIX (draft) specification of the AWK language [37].

F.1.1 Input Stage Definitions

During the input stage of the program execution, the user specifies the values of the design variables the program will later use to produce the fillet geometry, grid and Navier-Stokes flow solution. The specifications apper in the file `surface.options`. The user must provide the value or values of a single variable in one line, with the name of the variable followed by the assigned value of the variable. This format is shown below:

```
SPAN 0.213452
T/C-RATIO-UP 3.213453
Ct-Cf-RATIO 63.23
T/C-RATIO-LO 462.35
SCALEXY 3241.34  25643
```

Based on these input values, the program will produce the appropriate code and will insert it in the geometry definition file for the EAGLE program (See Appendix A). This is done separately for the four bounding lines of the fillet, i.e. the upper wing fillet-fuselage line, the upper wing fillet-wing line, the lower wing fillet-fuselage line and the lower wing fillet-wing line. The programs that achieve that are written in GAWK and are listed below.

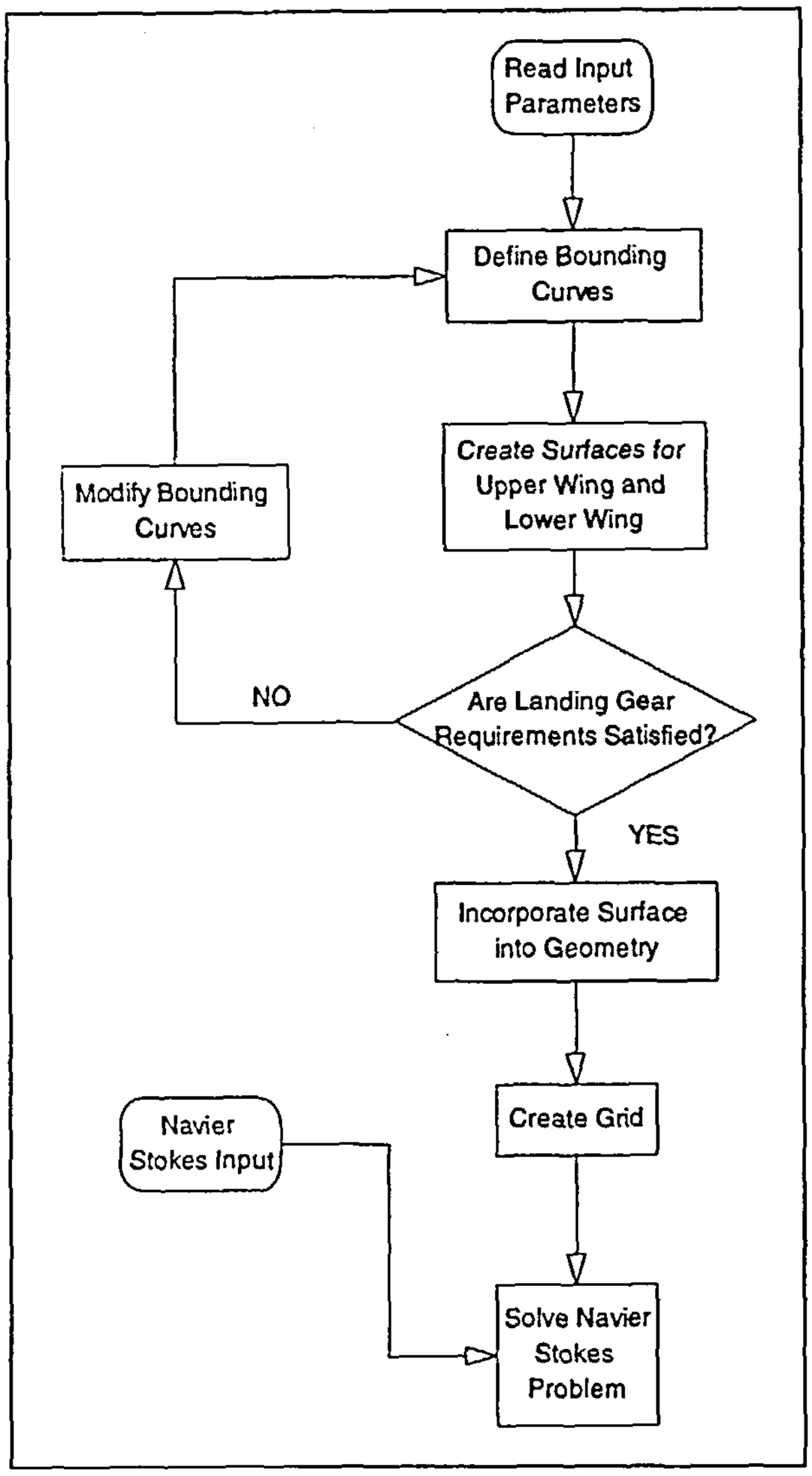

Figure F.1: Analysis Modules' Flow Chart

The first GAWK program creates the code that isolates the coordinates of the bounding lines described above from the aircraft geometry code. The code is written in four files, namely `isectu1.srf`, `isectu2.srf`, `isectl1.srf`, `isectl2.srf`, that will later be inserted in the appropriate locations in the geometry definition file for the aircraft.

```
BEGIN {
print "*   Entering data for ISECTU2" > "isectu2.srf"
print "*   Entering data for ISECTU1" > "isectu1.srf"
print "*   Entering data for ISECLO1" > "isectl1.srf"
print "*   Entering data for ISECLO2" > "isectl2.srf"
}
$1=="SPAN" {
span=$2
print"$'trans',corein=323,origin=-1000," span ",0,
     coreout=324$" >>"isectu2.srf"
print"$'intsec',male=300,female=324,
     coreout=1$" >> "isectu2.srf"
print"$'trans',corein=323,origin=-1000," span+300 ",0,
     coreout=324$" >>"isectu2.srf"
print"$'intsec',male=300,female=324,
     coreout=3$" >> "isectu2.srf"

print"$'trans',corein=323,origin=-1000," span ",0,
     coreout=324$">>"isectl2.srf"
print"$'intsec',male=301,female=324,
     coreout=2$">>"isectl2.srf"
print"$'trans',corein=323,origin=-1000," span+300 ",0,
     coreout=324$">>"isectl2.srf"
print"$'intsec',male=301,female=324,
     coreout=4$">>"isectl2.srf"
}

$1=="SCALEXZ" {
scalex=$2
scalez=$3
offsetx=-(scalex*9625-9625)
offsetz=scalez*2242-2242
offsetxd=-((scalex+.05)*9625-9625)
offsetzd=(scalez+.05)*2242-2242
```

```
print scalex, scalez, offsetx, offsetz, offsetxd, offsetzd
print"$'trans',corein=51,scale=" scalex ",1," scalez ",
      coreout=280$">>"isectu1.srf"
print"$'trans',corein=51,scale=" scalex+.05 ",1.0,
   " scalez+0.05 ",coreout=5$">>"isectu1.srf"
print"*">>"isectu1.srf"
print"*Correction for the offset by the z-scaling"
>>"isectu1.srf"
print"*                    i.e. (1.9 x 2242)- 2242"
>>"isectu1.srf"
print"*Correction for the offset by the x-scaling"
>>"isectu1.srf"
print"*                    i.e.  (1.9 x 9625) -9625"
>>"isectu1.srf"
print"$'trans',corein=280,origin=" offsetx ",0," offsetz ",
      coreout=280$">>"isectu1.srf"
print"$'trans',corein=5,origin=" offsetxd ",0," offsetzd ",
 coreout=5$">>"isectu1.srf"
print"$'trans',corein=280,origin=0,-9000,0,coreout=301$"
>>"isectu1.srf"
print"$'trans',corein=280,origin=0,9000,0,coreout=302$"
>>"isectu1.srf"
print"$'blend',bound=301,302,curves=30,coreout=303$"
>>"isectu1.srf"
print"$'intsec',male=303,female=597,coreout=304$"
>>"isectu1.srf"
print"$'trans',corein=5,origin=0,-9000,0,coreout=301$"
>>"isectu1.srf"
print"$'trans',corein=5,origin=0,9000,0,coreout=302$"
>>"isectu1.srf"
print"$'blend',bound=301,302,curves=30,coreout=303$"
>>"isectu1.srf"
print"$'intsec',male=303,female=597,coreout=5$"
>>"isectu1.srf"
print"$'output',corein=5,form='e',fileout=22,
  content='no',filnam='isectu1d.dat'$"
>>"isectu1.srf"
print"$'output',corein=304,form='e',fileout=16,
  content='no',filnam='isectu1.dat'$"
>>"isectu1.srf"
```

```
offsetxd=-((scalex-.05)*9625-9625)
print"$'trans',corein=50,scale=" scalex ",1,1.0,
   coreout=380$">>"isectl1.srf"
print"$'trans',corein=50,scale=" scalex-0.05 ",1,0.95,
   coreout=6$">>"isectl1.srf"
print"*">>"isectu2.srf"
print"*Correction for the offset by the z-scaling"
>>"isectl1.srf"
print"*                          i.e. (1.9 x 2242)- 2242"
>>"isectu2.srf"
print"*       Correction for the offset by the x-scaling"
>>"isectl1.srf"
print"*                          i.e.  (1.9 x 9500) -9500"
>>"isectl1.srf"
print"$'trans',corein=380,origin=" offsetx ",0,0,
   coreout=380$">>"isectl1.srf"
print"$'trans',corein=6,origin=" offsetxd ",0,-112.1,
   coreout=6$">>"isectl1.srf"
print"$'trans',corein=380,origin=0,-9000,0,
coreout=401$">>"isectl1.srf"
print"$'trans',corein=380,origin=0,9000,0,
coreout=402$">>"isectl1.srf"
print"$'blend',bound=401,402,curves=50,
coreout=403$">>"isectl1.srf"
print"$'intsec',male=403,female=201,
coreout=404$">>"isectl1.srf"
print"$'trans',corein=6,origin=0,-9000,0,
coreout=401$">>"isectl1.srf"
print"$'trans',corein=6,origin=0,9000,0,
coreout=402$">>"isectl1.srf"
print"$'blend',bound=401,402,curves=50,
coreout=403$">>"isectl1.srf"
print"$'intsec',male=403,female=201,
coreout=6$">>"isectl1.srf"
print"$'output',corein=6,form='e',fileout=23,
  content='no',filnam='isectl1d.dat'$"
>>"isectl1.srf"
print"$'output',corein=404,form='e',fileout=17,
  content='no',filnam='isectl1.dat'$"
>>"isectl1.srf"
```

```
}
```

Having created the code portions that will produce the coordinates of the bounding curves the input file specified, the following four programs incorporate the four files that were just created into the geometry definition file. The following file incorporates the fillet-fuselage bounding curve into the geometry definition file.

```
BEGIN {count=0
count1=0
print "*   ----------------------" > "test1.output"
print "*   TOUCHED BY ISECTL1.AWK" >> "test1.output"
print "*   ----------------------"  >> "test1.output"
print "*   ======================" > "test2.output"
print "*   INSERTION OF THE INTERSECTED
      CURVE ENDS HERE  " >> "test2.output"
print "*   ============ >> "test2.output"
}
$1!="*ISECTL1" && $1!="*ISECTL1END" && count==0
      {print $0 >> "test1.output"}
$1!="*ISECTL1" && $1!="*ISECTL1END" && count==2 {print $0 >>
"test2.output"}
$1!="*ISECTL1" && $1!="*ISECTL1END" && count==1 {}
$1=="*ISECTL1" { count=1
 count1=1
 print $0 >> "test1.output"}
$1=="*ISECTL1END"{count=2
print $0 >> "test2.output"}
END {
print "*   ============================" >> "test1.output"
print "*   INSERTION OF THE INTERSECTED
       CURVE BEGINS HERE" >> "test1.output"
print "* ============================" >> "test1.output"
system("cat test1.output isectl1.srf test2.output
> test.srf")
system("rm test1.output test2.output")
}
```

The remaining programs inocrporating the other three bounding curves into the geometry definition file have similar structures.

After all the bounding curve code segments are incorporated into the geometry definition file, the EAGLE surface generation code is executed and the actual coordinates of the bounding curves are obtained.

F.1.2 Fillet Surface Definition

The first step for creating a Bezier surface is to find the Bezier representation of the bounding curves. The programs presented here will implement the Higher Order (quartic) Bezier Curve approximation (See Appendix C) and will create a nonuniform Bezier surface with a control net of 5×4 Bezier points. The first routine listed is `program JOINBEZ`. It is a control routine which executes the quartic Bezier Curve approximations to each of the bounding curves, and then stores into a file the coordinates of the Bezier points of each of the four user-defined bounding curves.

```
program JOINBEZ
c       This routine puts together in a single file
c the coefficients for the
c       two bezier curves for the upper or
c lower surface and also generates
c       the coefficients for the remaining two curves.
c  The combined points
c       are stored in the file upbez.dat and
c lobez.dat for the upper surface
c       and the lower surface of the fillet
c
integer curv
double precision x1(5),x2(5),b1(5,3),b2(5,3)
DOUBLE PRECISION  BU1(5,3),BU2(5,3),BL1(5,3),BL2(5,3)

curv=11
write(6,*)'Entering Curve 11'
call coeffs(curv,BU1)
curv=12
write(6,*)'Entering Curve 12'
call coeffs(curv,BU2)
```

```
write(6,*)'Entering Curve 21'
curv=21
call coeffs(curv,BL1)
curv=22
write(6,*)'Entering Curve 22'
call coeffs(curv,BL2)
C
C       ARRAYS BU1 AND BU2 NOW HAVE THE
c COEFFICIENTS OF THE BEZIER CURVES
C       FOR THE UPPER SURFACE, WHILE ARRAYS
c BL1 AND BL2 HAVE THE COEFFICIENTS
C       OF THE BEZIER CURVES FOR THE LOWER SURFACE
C
call system('mv allcoeffs.dat allcoeffs.bak')
open(21,file='allcoeffs.dat',form='formatted',status='new')
do 10 i=1,5
write(21,100) bu1(i,1),bu1(i,2),bu1(i,3)
 10     continue
write(21,110)
write(21,110)
do 20 i=1,5
write(21,100) bu2(i,1),bu2(i,2),bu2(i,3)
 20     continue
write(21,110)
write(21,110)
do 30 i=1,5
write(21,100) bl1(i,1),bl1(i,2),bl1(i,3)
 30     continue
write(21,110)
write(21,110)
do 40 i=1,5
write(21,100) bl2(i,1),bl2(i,2),bl2(i,3)
 40     continue
close(21)
 100    format(3e20.8)
 110    format()
c       return
END
C
```

Program JOINBEZ calls the routine coeffs to determine the Bezier points that will produce a reasonable approximation to the bounding curves. Subroutine coeffs calls the optimizer NPSOL (See Appendix E) to minimize the tension surface (See Appendix C) between the Bezier curve and the original curve. In this implementation, no derivative information is given, and therefore the optimizer will apply finite differences to determine the gradient and jacobian information required for the optimization. The approximated tension surface is calculated in the subroutine OBJFN1. The subroutine coeffs is called four times, once for each bounding curve to be approximated. The outputs of the subroutine are the coordinates of the Bezier points that will produce the best approximating Bezier curve to the original bounding curve.

```
subroutine coeffs(curv,B)
c
c       curve data is in isect.dat
c       while bezier data is in data.dat
c
c       There are 5 coefficients in this implementation
c with 3 components each. Of these, only
c       the 3 inner coefficients are required to change,
c  giving thus 9
c       variables to optimize.
c
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

      IMPLICIT           DOUBLE PRECISION(A-H,O-Z)

*     Set the declared array dimensions.
*     NROWA  = the declared row dimension of  A.
*     NROWJ  = the declared row dimension of  CJAC.
*     NROWR  = the declared row dimension of  R.
*     MAXN   = maximum no. of variables allowed for.
*     MAXBND = maximum no. of variables+lin&nonlin constr.
*     LIWORK = the length of the integer work array.
*     LWORK  = the length of the double precision work array.
```

```
      PARAMETER          (NROWA=2, NROWJ=0, NROWR =30,
     $                    MAXN=5, LIWORK = 20, LWORK = 200,
     $                    MAXBND=MAXN + NROWA + NROWJ)

      INTEGER            ISTATE(MAXBND)
      INTEGER            IWORK(LIWORK),curv,curtyp
      DOUBLE PRECISION   A(NROWA,MAXN)
      DOUBLE PRECISION   BL(MAXBND), BU(MAXBND)
      DOUBLE PRECISION   C(NROWJ), CJAC(NROWJ,MAXN),
      DOUBLE PRECISION   CLAMDA(MAXBND)
      DOUBLE PRECISION   OBJGRD(MAXN), R(NROWR,MAXN), X(MAXN)
      DOUBLE PRECISION   WORK(LWORK),LOAD(60),SPLINE(60)
      DOUBLE PRECISION   ABY, BBY, ABZ, BBZ, AEY, AEZ, BEY,BEZ
      DOUBLE PRECISION   BU1(5,3),BU2(5,3),BL1(5,3),BL2(5,3)
      double precision   b(5,3)
      double precision  tenth,fourth,nine_10,three_4,half
c      common /MYPARAMS/ ABY,BBY,ABZ,BBZ,AEY,
     $                  BEY,AEZ,BEZ,BU1,BU2,
c     %                    BL1, BL2
common /MYPARAMS/ ABY,BBY,ABZ,BBZ,AEY,BEY,AEZ,BEZ
common /DESCRIPT/ curtyp
      EXTERNAL           OBJFN1,CONFN1
      PARAMETER          (ZERO = 0.0, ONE = 1.0)

*     Set the actual problem dimensions.
*     N      = the number of variables.
*     NCLIN  = the number of general lin constr (may be 0).
*     NCNLN  = the number of nonlin constr (may be 0).

      N      = 5
      NCLIN  = 2
      NCNLN  = 0
      NBND   = N + NCLIN + NCNLN

*     ------------------------------------------------------
*     Assign file numbers and the data arrays.
*     NOUT   = the unit number for printing.
*     IOPTNS = the unit number for reading the options file.
*     Bounds  .ge.    BIGBND  treated as plus  infinity.
*     Bounds  .le.  - BIGBND  Treated as minus infinity.
*     A      = the linear constraint matrix.
*     BL     = the lower bounds on  x,  a'x  and  c(x).
```

```
*     BU     = the upper bounds on  x,  a'x  and  c(x).
*     X      = the initial estimate of the solution.
*     ----------------------------------------------------
      NOUT   = 31
      call system('rm output.npsol')
      open(31,file='output.npsol',status='new')
      open(30,file='options.npsol',status='old')
      IOPTNS = 30
      BIGBND = 1.0D+15
      curtyp=curv

*     Set the matrix  A.

       DO 40 J = 1, N
  DO 30 I = 1, NCLIN
     if (i.ne.j)A(I,J) = ZERO
     if (i.eq.j)A(i,j) = ONE
   30    CONTINUE
   40 CONTINUE
*
*       Do some file reading
*
if (curv.eq.11) then
open(4,file='isectu1.dat',form='formatted',status='old')
do 410 i=1,20
l=(i-1)*3
read(4,420)SPLINE(l+1),SPLINE(l+2),SPLINE(l+3)
c                write(6,*)'reading isectu1'
 410    continue
 420    format(3e20.8)
close(4)
elseif (curv.eq.12) then
open(4,file='isectu2.dat',form='formatted',status='old')
do 430 i=1,20
l=(i-1)*3
read(4,420)SPLINE(l+1),SPLINE(l+2),SPLINE(l+3)
c                write(6,*)'Reading isectu2'
 430    continue
close(4)
elseif (curv.eq.21) then
open(4,file='isectl1.dat',form='formatted',status='old')
do 440 i=1,20
```

```
l=(i-1)*3
read(4,420)SPLINE(l+1),SPLINE(l+2),SPLINE(l+3)
write(6,*)'Reading isectl1'
 440    continue
close(4)
elseif (curv.eq.22) then
open(4,file='isectl2.dat',form='formatted',status='old')
do 450 i=1,20
l=(i-1)*3
read(4,420)SPLINE(l+1),SPLINE(l+2),SPLINE(l+3)
c                write(6,*)'Reading isectl2'
 450    continue
close(4)
else
write(6,*) 'Curve coordinate specs are incorrect.'
stop
endif

C*******************************************************
C      DEFINE THE MOST IMPORTANT COMMON BLOCK PARAMETERS
C*******************************************************
C
ABY=(SPLINE(5)-SPLINE(2))/(SPLINE(4)-SPLINE(1))
BBY=SPLINE(2)-(SPLINE(5)-SPLINE(2))/(SPLINE(4)-SPLINE(1))
     %   *SPLINE(1)
AEY=(SPLINE(59)-SPLINE(56))/(SPLINE(58)-SPLINE(55))
BEY=SPLINE(56)-(SPLINE(59)-SPLINE(56))/
     %           (SPLINE(58)-SPLINE(55))*SPLINE(55)

ABZ=(SPLINE(6)-SPLINE(3))/(SPLINE(4)-SPLINE(1))
BBZ=SPLINE(3)-(SPLINE(6)-SPLINE(3))/(SPLINE(4)-SPLINE(1))
     %   *SPLINE(1)
AEZ=(SPLINE(60)-SPLINE(57))/(SPLINE(58)-SPLINE(55))
BEZ=SPLINE(57)-(SPLINE(60)-SPLINE(57))/
     %           (SPLINE(58)-SPLINE(55))*SPLINE(55)

C
C*******************************************************
c       open(3,file='data.dat',form='formatted',status='old')
c       do 180 i=1,20
```

```
c                 j=(i-1)*3
c                 read(3,190) load(j+1),load(j+2),load(j+3)
c 180    continue
c 190    format(3e20.8)
c        close(3)
*      Set the initial estimate of  X.
tenth=spline(1)+(spline(58)-spline(1))/10.00
half=(spline(1)+spline(58))/2.00
nine_10=spline(1)+9.00*(spline(58)-spline(1))/10.00
three_4=spline(1)+3.00*(spline(58)-spline(1))/4.00
fourth=spline(1)+(spline(58)-spline(1))/4.00

       A(1,1) = -ONE
       A(1,2) = ONE
       A(2,2) = -ONE
       A(2,5) = ONE



       X(1)    =  tenth
       X(2)    =  fourth
       X(3)    =  SPLINE(23)
       X(4)    =  SPLINE(24)
       X(5)    =  three_4

*
*      Set the bounds.
*
       DO 50 J  =  1, NBND
 BL(J) = -BIGBND
 BU(J) =  BIGBND
   50 CONTINUe
        BL(1)  =  SPLINE(1)
        BL(2)  =  spline(1)
        BL(5)  =  half
        BU(1)  =  fourth
        BU(2)  =  nine_10
        BU(5)  =  nine_10
*
*        ideal BL for upper1 section 1000 and 4000
*        ideal BL for lower1 section 1000 and 1000
*
```

```
IF (CURV.EQ.11) THEN
       BL(6)  =  1000
       BL(7)  =  4000
*
*       Ideal BU for upper1 sections 5000 and 7000
*       ideal BU for lower1 sections 7000 and 7000
*
       BU(6) =   5000
       BU(7) =   7000

      ELSEIF (CURV.EQ.12) THEN
      BL(3) = spline(2)
      BL(6) = 1000
      BL(7) = 4000
      BU(3) = spline(59)
      BU(6) = 5000
      BU(7) = 7000

      ELSEIF (CURV.EQ.22) THEN
      BL(1) = spline(10)
      BL(2) = spline(31)
      BL(3) = SPLINE(2)
      BL(5) = spline(40)
      BL(6) = 1000
      BL(7) = 1000
      BU(3) = SPLINE(59)
      BU(4) = spline(33)*1.2
      BU(5) = spline(58)
      BU(6) = 7000
      BU(7) = 7000
      write(6,*)'Detected curv 22'

      ELSE IF (CURV.EQ.21) THEN
      c               BL(1) = spline(10)
      c               BL(2) = spline(30)
      c               BL(3) = -bigbnd
      c               BL(4) = -bigbnd
      c               BL(5) = half
      BL(6) = 1000
      BL(7) = 1000
      c               BU(2)= spline(40)
      c               BU(3) = -1000
```

```
c                BU(4) = spline(27)
c                BU(1) = spline(25)
BU(6) = 7000
BU(7) = 7000
ENDIF

*     ------------------------------------------
*     Read the options file.
*     ------------------------------------------

      CALL NPFILE( IOPTNS, INFORM )
      IF (INFORM .NE. 0) THEN
 WRITE (NOUT, 3000) INFORM
 STOP
      END IF

*     ------------------------------------------
*     Solve the problem.
*     ------------------------------------------
write(6,*)'Entering NPSOL'
      CALL NPSOL ( N, NCLIN, NCNLN, NROWA, NROWJ, NROWR,
     $             A, BL, BU,
     $             CONFN1, OBJFN1,
     $             INFORM, ITER, ISTATE,
     $             C, CJAC, CLAMDA, OBJF, OBJGRD, R, X,
     $             IWORK, LIWORK, WORK, LWORK )
*       write(6,*)'Just exited NPSOL'
       IF (INFORM .GT. 0) GO TO 900

*     -----------
*     Error exit.
*     -----------

  900 WRITE (NOUT, 3010) INFORM
if (curv.eq.11) then
BU1(1,1)=spline(1)
BU1(1,2)=spline(2)
BU1(1,3)=spline(3)
BU1(2,1)=x(1)
BU1(2,2)=ABY*X(1)+BBY
BU1(2,3)=ABZ*X(1)+BBZ
BU1(3,1)=X(2)
```

```
BU1(3,2)=X(3)
BU1(3,3)=X(4)
BU1(4,1)=X(5)
BU1(4,2)=X(5)*AEY+BEY
BU1(4,3)=X(5)*AEZ+BEZ
BU1(5,1)=SPLINE(58)
BU1(5,2)=SPLINE(59)
BU1(5,3)=SPLINE(60)
write(6,*),bu1(1,1),bu1(5,1)
ELSEIF (CURV.EQ.12) THEN
BU2(1,1)=spline(1)
BU2(1,2)=spline(2)
BU2(1,3)=spline(3)
BU2(2,1)=x(1)
BU2(2,2)=ABY*X(1)+BBY
BU2(2,3)=ABZ*X(1)+BBZ
BU2(3,1)=X(2)
BU2(3,2)=X(3)
BU2(3,3)=X(4)
BU2(4,1)=X(5)
BU2(4,2)=X(5)*AEY+BEY
BU2(4,3)=X(5)*AEZ+BEZ
BU2(5,1)=SPLINE(58)
BU2(5,2)=SPLINE(59)
BU2(5,3)=SPLINE(60)
ELSEIF (CURV.EQ.21) THEN
BL1(1,1)=spline(1)
BL1(1,2)=spline(2)
BL1(1,3)=spline(3)
BL1(2,1)=x(1)
BL1(2,2)=ABY*X(1)+BBY
BL1(2,3)=ABZ*X(1)+BBZ
BL1(3,1)=X(2)
BL1(3,2)=X(3)
BL1(3,3)=X(4)
BL1(4,1)=X(5)
BL1(4,2)=X(5)*AEY+BEY
BL1(4,3)=X(5)*AEZ+BEZ
BL1(5,1)=SPLINE(58)
BL1(5,2)=SPLINE(59)
BL1(5,3)=SPLINE(60)
ELSEIF (CURV.EQ.22) THEN
```

```
BL2(1,1)=spline(1)
BL2(1,2)=spline(2)
BL2(1,3)=spline(3)
BL2(2,1)=x(1)
BL2(2,2)=ABY*X(1)+BBY
BL2(2,3)=ABZ*X(1)+BBZ
BL2(3,1)=X(2)
BL2(3,2)=X(3)
BL2(3,3)=X(4)
BL2(4,1)=X(5)
BL2(4,2)=X(5)*AEY+BEY
BL2(4,3)=X(5)*AEZ+BEZ
BL2(5,1)=SPLINE(58)
BL2(5,2)=SPLINE(59)
BL2(5,3)=SPLINE(60)
ENDIF
c       write(6,*)'Solution vector is ',(x(i),i=1,maxn)
c
c       open(15,file='coeffs.dat',form='formatted',status='new')
c               write(15,1999) spline(1),spline(2),spline(3)
c               write(15,1999) x(1),ABY*x(1)+BBY,ABZ*(x1)+BBZ
c               write(15,1999) x(2),x(3),x(4)
c               write(15,1999) x(5),AEY*x(5)+BEY,AEZ*x(5)+BEZ
c       close(15)
c 1999  format(3e16.8)
C      STOP

 3000 FORMAT(/ ' NPFILE terminated with  INFORM =', I3)
 3010 FORMAT(/ ' NPSOL  terminated with  INFORM =', I3)

*     End of the example program for NPSOL.
close(30)
close(31)
if (curv.eq.11) then
do 300 i=1,5
do 300 j=1,3
b(i,j)=bu1(i,j)
 300              continue
elseif (curv.eq.12) then
do 310 i=1,5
do 310 j=1,3
b(i,j)=bu2(i,j)
```

```
 310              continue
elseif (curv.eq.21) then
do 320 i=1,5
do 320 j=1,3
b(i,j)=bl1(i,j)
 320              continue
elseif (curv.eq.22) then
do 330 i=1,5
do 330 j=1,3
b(i,j)=bl2(i,j)
 330              continue
endif
return

      END
*+++++++++++++++++++++++++++++++++++++++++++++++

      SUBROUTINE OBJFN1(MODE,N,X,OBJF,OBJGRD,NSTATE)
      IMPLICIT             DOUBLE PRECISION(A-H,O-Z)
      DOUBLE PRECISION   X(N), OBJGRD(N),LOAD1(60)
      DOUBLE PRECISION  LOAD2(60),EDGPTS(6)
      DOUBLE PRECISION   area,A(3),B(3),C(3)
      INTEGER   CURV

      common /DESCRIPT/ curv

*-----------------------------------------
*     OBJFN1  computes the value and first
*     derivatives of the nonlinear
*     objective function.
*-----------------------------------------
if (curv.eq.11) then
open(3,file='isectu1.dat',form='formatted',status='old')
elseif (curv.eq.12) then
open(3,file='isectu2.dat',form='formatted',status='old')
elseif (curv.eq.21) then
open(3,file='isectl1.dat',form='formatted',status='old')
elseif (curv.eq.22) then
open(3,file='isectl2.dat',form='formatted',status='old')
endif
do 10 i=1,20
l=(i-1)*3
```

```
read(3,20)load1(l+1),load1(l+2),load1(l+3)
 10      continue
 20      format(3e20.8)
close(3)
c        write(6,*)'Checkpoint'
edgpts(1)=load1(1)
edgpts(2)=load1(2)
edgpts(3)=load1(3)
edgpts(4)=load1(58)
edgpts(5)=load1(59)
edgpts(6)=load1(60)
c
c       Use the coeffs to create the file data.dat from horner.f
c
call horner(x,edgpts)
c
c       Read in the new bezier data
c
open(4,file='data.dat',form='formatted',status='old')
do 30 i=1,20
l=(i-1)*3
read(4,40)load2(l+1),load2(l+2),load2(l+3)
30      continue
40      format(3e20.8)
close(4)
c       OBJF = 0.00
c       do 50 i=1,60
c        OBJF=OBJF+abs(load1(i)-load2(i))**2
c 50    continue


*
*       Define the new objective as the sum
*  of the areas of the triangles
*       forming the surface between the two curves.
*
area=0.0
OBJF=0.0
do 50 i=1,36
if (i.eq.1) then
A(1)=LOAD1(1)
A(2)=LOAD1(2)
```

```
A(3)=LOAD1(3)
B(1)=LOAD1(4)
B(2)=LOAD1(5)
B(3)=LOAD1(6)
C(1)=LOAD2(4)
C(2)=LOAD2(5)
C(3)=LOAD2(6)
call trarea(A,B,C,area)
else if (i.eq.36) then
A(1)=LOAD1(55)
A(2)=LOAD1(56)
A(3)=LOAD1(57)
B(1)=LOAD2(55)
B(2)=LOAD2(56)
B(3)=LOAD2(57)
C(1)=LOAD1(58)
C(2)=LOAD1(59)
C(3)=LOAD1(60)
call trarea(A,B,C,area)
   else if (abs((i*1.0)/2.0-int((i*1.0)/2.0)).gt.0.0001) then
l=(((i+1)/2)-1)*3+1
A(1)=LOAD2(l)
A(2)=LOAD2(l+1)
A(3)=LOAD2(l+2)
l=((i+1)/2)*3+1
B(1)=LOAD2(l)
B(2)=LOAD2(l+1)
B(3)=LOAD2(l+2)
C(1)=LOAD1(l)
C(2)=LOAD1(l+1)
C(3)=LOAD1(l+2)
call trarea(A,B,C,area)
c write(6,*)'Areao ',i,' vertices ',((i+1)/2-1)*3+1,l,l
c write(6,*)'Points ',(i+1)/2,(i+1)/2+1,(i+1)/2+1
  else if (abs((i*1.0)/2.0-int((i*1.0)/2.0)).lt.0.0001) then
l=((i/2+2)-1)*3+1
A(1)=LOAD2(l)
A(2)=LOAD2(l+1)
A(3)=LOAD2(l+2)
l=((i/2+2)-2)*3+1
B(1)=LOAD2(l)
B(2)=LOAD2(l+1)
```

```
B(3)=LOAD2(1+2)
C(1)=LOAD1(1)
C(2)=LOAD1(1+1)
C(3)=LOAD1(1+2)
call trarea(A,B,C,area)
c write(6,*)'Areae ',i,' vertices ',((i/2+2)-1)*3+1,1,1
c write(6,*)'points ',i/2+2,i/2+2-1,i/2+2-1
endif
OBJF=OBJF+sqrt(area**2)
 50     continue
*       write(6,*)'OBJ value=',OBJF
*      OBJGRD(1) =   -X(2)*X(3)
*      OBJGRD(2) =   -X(1)*X(3)
*      OBJGRD(3) =   -X(1)*X(2)
       RETURN

*      End of  OBJFN1.

       END
*************************************************
subroutine trarea(AA,BB,CC,area)
c
c       calculates the area of the triangle given
C the three vertices
c       of the triangle.
c
 double precision AA(3),BB(3),CC(3),a(3)
 DOUBLE PRECISION b(3),area, f1,f2,f3,f4
c
c       derive the first and second vector
c
a(1)=CC(1)-AA(1)
a(2)=CC(2)-AA(2)
a(3)=CC(3)-AA(3)
b(1)=BB(1)-AA(1)
b(2)=BB(2)-AA(2)
b(3)=BB(3)-AA(3)

c
c use the cross product definition for
c the area of a parallelogram
```

```
c       to derive the area of the triangle
c using the two vectos a and b
c
f1=a(2)*b(3)-b(2)*a(3)
f2=a(1)*b(3)-a(3)*b(1)
f3=a(1)*b(2)-a(2)*b(1)

f3=f1**2+f2**2+f3**2
f4=sqrt(f3)
area=f4/2.00
return
end
```

Subroutine coeffs in turn calls subroutine horner to compute one coordinate of a Bezier curve point. Of course, to compute all three coordinates of such a point the horner routine would have to be called three times. The subroutine uses a Horner-like scheme of nested multiplications[53] to compute one coordinate value of a Bezier curve which is more efficient than the de Casteljau algorithm both in terms of memory requirements and also in terms of efficiency[2].

```
 subroutine horner(X,edgpts)
c
c
c
integer i,j,k,degree,npoints
double precision coeff(0:4),points(0:100),xpts(0:100)
double precision ypts(0:100),bx(0:100),by(0:100),bz(0:100)
double precision zpts(0:100),x(9),edgpts(6)
double precision ABY,BBY,ABZ,BBZ,AEY,BEY,AEZ,BEZ

common /MYPARAMS/ ABY,BBY,ABZ,BBZ,AEY,BEY,AEZ,BEZ

degree=4
npoints=19
coeff(0)=edgpts(1)
```

[2]The de Casteljau algoritm would require the use of an auxiliary array to store the curve coefficients which would have to be updated for each function call

```
coeff(1)=x(1)
coeff(2)=x(2)
coeff(3)=x(5)
coeff(4)=edgpts(4)
do 10 i=0,degree
 10                bx(i)=coeff(i)
call bezpt(degree,npoints,coeff,xpts)
coeff(0)=edgpts(2)
coeff(1)=ABY*X(1)+BBY
coeff(2)=x(3)
coeff(3)=AEY*X(5)+BEY
coeff(4)=edgpts(5)
do 20 i=0,degree
 20                by(i)=coeff(i)
call bezpt(degree,npoints,coeff,ypts)
coeff(0)=edgpts(3)
coeff(1)=ABZ*X(1)+BBZ
coeff(2)=x(4)
coeff(3)=AEZ*X(5)+BEZ
coeff(4)=edgpts(6)
do 30 i=0,degree
 30                bz(i)=coeff(i)
call bezpt(degree,npoints,coeff,zpts)
call system('mv data.dat data.bak')
open (4,file='data.dat',form='formatted',status='new')
do 40 i=0,npoints
write(4,5)xpts(i),ypts(i),zpts(i)
c                 write(6,5)xpts(i),ypts(i),zpts(i)
40      continue
5       format(3e20.8)
c       write(6,*)' '
do 50 i=0,degree
write(4,5)bx(i),by(i),bz(i)
c                 write(6,5)bx(i),by(i),bz(i)
50      continue
close(4)
end
subroutine bezpt(degree, npoints, coeff,points)
c
c       Converts Bezier curve into point sequence.
c       Input:  degree:  degree of curve
c       npoints:  number of coordinates to be generated.
```

```
c       coeff:  coordinates of control polygon.
c       output:  points:  coordinate of points on curve
c Modified code originally presented in Farin[1990]
integer degree, npoints
double precision coeff(0:10), points(0:100)
double precision fload, t, delt
integer i
double precision coord

delt=1.0/(npoints*1.0)
t=0.0
do 10 i=0,npoints
call hornbez(degree,coeff,t,points(i))
t=t+delt
 10     continue
return
end

subroutine hornbez(degree,coeff,t,coord)
c
c       Uses a Hoerner like scheme to compute
c  one coordinate value of a
c       Bezier Curve.  Has to be called for
c       each coordinate (x,y,z)
c       of a control polygon.
c       Input:   degree: degree of curve
c                coeff:  Array with coefficients of curve
c                t:      parameter value
c       Output:  coord:  coordinate value
c This code is a modified version of the one
c  presented in Farin[1990]
double precision coeff(0:10),t,coord,fact,t1,aux
integer degree, r,i,nchi

t1 = 1.0 -t
fact = 1.0

nchi=1
aux=coeff(0)*t1
do 10 i=1,degree-1
fact=fact*t
nchi=nchi*(degree-i+1)/i
```

```
aux=(aux+fact*nchi*coeff(i))*t1
 10      continue
aux=aux+fact*t*coeff(degree)
coord=aux
return
end
```

By the completion of the JOINBEZ routine, the coefficients of the bounding curves are calculated and the program then proceeds to the calculation of the actual bezier surfaces. This is done in routine testplot. The routine assigns initial values to the Bezier points of the surface control net, and calculates the coordinates of the surface points at a given curve resolution, in the case shown below six curves of twenty points each.

```
program testplot

c
c       requires files plotsur.f and horner.f

double precision bxu(21,21),byu(21,21),bzu(21,21)
double precision bxl(21,21),byl(21,21),bzl(21,21)
double precision bxu1d(20),byu1d(20),bzu1d(20)
double precision bxu2d(20),byu2d(20),bzu2d(20)
double precision bxl1d(20),byl1d(20),bzl1d(20)
double precision bxl2d(20),byl2d(20),bzl2d(20)
double precision bxu1(20),byu1(20),bzu1(20)
double precision bxu2(20),byu2(20),bzu2(20)
double precision bxl1(20),byl1(20),bzl1(20)
double precision bxl2(20),byl2(20),bzl2(20)
double precision u1(5,3),u2(5,3),l1(5,3),l2(5,3)
double precision factor1,factor2,proport
integer i,j,k,curves

open(30,file='surface.options',form='formatted',status='old')
read(30,600)factor1
read(30,605)factor2
read(30,607)factor3
read(30,610)curves
```

```
close(30)
 proport=.3333
c
c       controls the steepness of the fillet-fuselage junction
c
 600    format('FACTOR1',1x,f8.4)
c
c       controls the steepness of the fillet-wing junction
c
 605    format('FACTOR2',1x,f8.4)
c
c       controls the steepness of the leading and trailing edge of the
c       fillet
c
 607    format('FACTOR3',1x,f8.4)
 610    format('CURVES',1x,i3)
write(6,600)factor1
write(6,605)factor2
write(6,607)factor3
factor=10.00
open(3,file='allcoeffs.dat',form='formatted',status='old')
do 10 i=1,5
read(3,120)bxu(i,1),byu(i,1),bzu(i,1)
 10     continue
read(3,130)
read(3,130)
do 20 i=1,5
read(3,120)bxu(i,4),byu(i,4),bzu(i,4)
c                          write(6,120)bxu(i,4),byu(i,4),bzu(i,4)
 20     continue
read(3,130)
read(3,130)
do 15 i=1,5
read(3,120)bxl(i,1),byl(i,1),bzl(i,1)
 15     continue
read(3,130)
read(3,130)
do 25 i=1,5
read(3,120)bxl(i,4),byl(i,4),bzl(i,4)
c                          write(6,120)bxl(i,4),byl(i,4),bzl(i,4)
 25     continue
close(3)
```

```
do 30 i=1,5
bxu(i,2)=(bxu(i,4)-bxu(i,1))*proport+bxu(i,1)
bxu(i,3)=(bxu(i,4)-bxu(i,1))*(1-proport)+bxu(i,1)
byu(i,2)=(byu(i,4)-byu(i,1))*proport+byu(i,1)
byu(i,3)=(byu(i,4)-byu(i,1))*(1-proport)+byu(i,1)
bzu(i,2)=(bzu(i,4)-bzu(i,1))*proport+bzu(i,1)
bzu(i,3)=(bzu(i,4)-bzu(i,1))*(1-proport)+bzu(i,1)
 30     continue
do 35 i=1,5
bxl(i,2)=(bxl(i,4)-bxl(i,1))*proport+bxl(i,1)
bxl(i,3)=(bxl(i,4)-bxl(i,1))*(1-proport)+bxl(i,1)
byl(i,2)=(byl(i,4)-byl(i,1))*proport+byl(i,1)
byl(i,3)=(byl(i,4)-byl(i,1))*(1-proport)+byl(i,1)
bzl(i,2)=(bzl(i,4)-bzl(i,1))*proport+bzl(i,1)
bzl(i,3)=(bzl(i,4)-bzl(i,1))*(1-proport)+bzl(i,1)
 35     continue

c       Read in derivatives

open (10,file='isectu1d.dat',form='formatted',status='old')
read(10,120)(bxu1d(i),byu1d(i),bzu1d(i),i=1,20)
close(10)
open (11,file='isectu2d.dat',form='formatted',status='old')
read(11,120)(bxu2d(i),byu2d(i),bzu2d(i),i=1,20)
close(11)
open (12,file='isectl1d.dat',form='formatted',status='old')
read(12,120)(bxl1d(i),byl1d(i),bzl1d(i),i=1,20)
close(12)
open (13,file='isectl2d.dat',form='formatted',status='old')
read(13,120)(bxl2d(i),byl2d(i),bzl2d(i),i=1,20)
close(13)
c
c       Read in boundary curves
c
open (14,file='isectu1.dat',form='formatted',status='old')
read(14,120)(bxu1(i),byu1(i),bzu1(i),i=1,20)
close(14)
open (15,file='isectu2.dat',form='formatted',status='old')
read(15,120)(bxu2(i),byu2(i),bzu2(i),i=1,20)
close(15)
open (16,file='isectl1.dat',form='formatted',status='old')
read(16,120)(bxl1(i),byl1(i),bzl1(i),i=1,20)
```

```
close(16)
open (17,file='isectl2.dat',form='formatted',status='old')
read(17,120)(bxl2(i),byl2(i),bzl2(i),i=1,20)
close(17)
c
c       Assign the derivatives
c
c       Lateral derivatives at root
c
t=abs(bxu2(1)-bxu1(1))/factor3+bxu1(1)
xa=bxu1d(1)
xb=bxu1(1)
ya=byu1d(1)
yb=byu1(1)
za=bzu1d(1)
zb=bzu1(1)
bxu(1,2)=t
byu(1,2)=(yb-ya)/(xb-xa)*(t-bxu(1,1))+byu(1,1)
bzu(1,2)=(zb-za)/(xb-xa)*(t-bxu(1,1))+bzu(1,1)
t=bxu1(20)-abs(bxu2(20)-bxu1(20))/factor3
xa=bxu1d(20)
xb=bxu1(20)
ya=byu1d(20)
yb=byu1(20)
za=bzu1d(20)
zb=bzu1(20)
bxu(5,2)=t
byu(5,2)=(yb-ya)/(xb-xa)*(t-bxu(5,1))+byu(5,1)
bzu(5,2)=(zb-za)/(xb-xa)*(t-bxu(5,1))+bzu(5,1)
c
c       Lateral derivatives at fillet-span
c
t=byu2(1)-abs(byu2(1)-byu1(1))/factor3
xa=bxu2d(1)
xb=bxu2(1)
ya=byu2d(1)
yb=byu2(1)
za=bzu2d(1)
zb=bzu2(1)
bxu(1,3)=((xb-xa)/(yb-ya))*(t-byu(1,4))+bxu(1,4)
byu(1,3)=t
bzu(1,3)=((zb-za)/(yb-ya))*(t-byu(1,4))+bzu(1,4)
```

```
t=byu2(20)-abs(byu2(20)-byu1(20))/factor3
xa=bxu2d(20)
xb=bxu2(20)
ya=byu2d(20)
yb=byu2(20)
za=bzu2d(20)
zb=bzu2(20)
bxu(5,3)=((xb-xa)/(yb-ya))*(t-byu(5,4))+bxu(5,4)
byu(5,3)=t
bzu(5,3)=((zb-za)/(yb-ya))*(t-byu(5,4))+bzu(5,4)
c
c       Derivatives at fillet-span
c
t=byu2(4)-abs(byu2(4)-byu1(4))/factor2
xa=bxu2d(4)
xb=bxu2(4)
ya=byu2d(4)
yb=byu2(4)
za=bzu2d(4)
zb=bzu2(4)
bxu(2,3)=((xb-xa)/(yb-ya))*(t-byu(2,4))+bxu(2,4)
byu(2,3)=t
bzu(2,3)=((zb-za)/(yb-ya))*(t-byu(2,4))+bzu(2,4)
t=byu2(9)-abs(byu2(9)-byu1(9))/factor2
c       xa=bxu2d(9)
xb=bxu2(9)
xa=bxu2(9)
ya=byu2d(9)
yb=byu2(9)
za=bzu2d(9)
zb=bzu2(9)
bxu(3,3)=((xb-xa)/(yb-ya))*(t-byu(3,4))+bxu(3,4)
byu(3,3)=t
bzu(3,3)=((zb-za)/(yb-ya))*(t-byu(3,4))+bzu(3,4)
t=byu2(16)-abs(byu2(16)-byu1(16))/factor2
xa=bxu2d(16)
xb=bxu2(16)
ya=byu2d(16)
yb=byu2(16)
za=bzu2d(16)
zb=bzu2(16)
```

```
bxu(4,3)=((xb-xa)/(yb-ya))*(t-byu(4,4))+bxu(4,4)
byu(4,3)=t
bzu(4,3)=((zb-za)/(yb-ya))*(t-byu(4,4))+bzu(4,4)
c
c       Derivatives at fillet root
c
t=bzu1(4)-abs(bzu2(4)-bzu1(4))/factor1
xb=bxu1(4)
xa=bxu1d(4)
yb=byu1(4)
ya=byu1d(4)
zb=bzu1(4)
za=bzu1d(4)
bxu(2,2)=(xb-xa)/(zb-za)*(t-bzu(2,1))+bxu(2,1)
byu(2,2)=(yb-ya)/(zb-za)*(t-bzu(2,1))+byu(2,1)
bzu(2,2)=t
t=bzu1(9)-abs(bzu1(9)-bzu2(9))/factor1
xb=bxu1(9)
xa=bxu1d(9)
yb=byu1(9)
ya=byu1d(9)
zb=bzu1(9)
za=bzu1d(9)
bxu(3,2)=(xb-xa)/(zb-za)*(t-bzu(3,1))+bxu(3,1)
byu(3,2)=(yb-ya)/(zb-za)*(t-bzu(3,1))+byu(3,1)
bzu(3,2)=t
t=bzu1(16)-abs(bzu1(16)-bzu2(16))/factor1
xb=bxu1(16)
xa=bxu1d(16)
yb=byu1(16)
ya=byu1d(16)
zb=bzu1(16)
za=bzu1d(16)
bxu(4,2)=(xb-xa)/(zb-za)*(t-bzu(3,1))+bxu(3,1)
byu(4,2)=(yb-ya)/(zb-za)*(t-bzu(3,1))+byu(3,1)
bzu(4,2)=t
c
c-----------------------------------------------
c
c       LOWER SURFACE OF THE FILLET
c-----------------------------------------------
c
```

```
c       Assign the derivatives
c
c       Lateral derivatives at root
c

t=abs(bxl2(1)-bxl1(1))/factor1+bxl1(1)
xb=bxl1d(1)
xa=bxl1(1)
yb=byl1d(1)
ya=byl1(1)
zb=bzl1d(1)
za=bzl1(1)
bxl(1,2)=t
byl(1,2)=(yb-ya)/(xb-xa)*(t-bxl(1,1))+byl(1,1)
bzl(1,2)=(zb-za)/(xb-xa)*(t-bxl(1,1))+bzl(1,1)
t=bxl1(20)-abs(bxl2(20)-bxl1(20))/factor1
xb=bxl1d(20)
xa=bxl1(20)
yb=byl1d(20)
ya=byl1(20)
zb=bzl1d(20)
za=bzl1(20)
bxl(5,2)=t
byl(5,2)=(yb-ya)/(xb-xa)*(t-bxl(5,1))+byl(5,1)
bzl(5,2)=(zb-za)/(xb-xa)*(t-bxl(5,1))+bzl(5,1)
c
c       Lateral derivatives at fillet-span
c
t=byl2(1)-abs(byl2(1)-byl1(1))/factor2
xa=bxl2d(1)
xb=bxl2(1)
ya=byl2d(1)
yb=byl2(1)
za=bzl2d(1)
zb=bzl2(1)
bxl(1,3)=((xb-xa)/(yb-ya))*(t-byl(1,4))+bxl(1,4)
byl(1,3)=t
bzl(1,3)=((zb-za)/(yb-ya))*(t-byl(1,4))+bzl(1,4)

t=byl2(20)-abs(byl2(20)-byl1(20))/factor2
xa=bxl2d(20)
xb=bxl2(20)
```

```
ya=byl2d(20)
yb=byl2(20)
za=bzl2d(20)
zb=bzl2(20)
bxl(5,3)=((xb-xa)/(yb-ya))*(t-byl(5,4))+bxl(5,4)
byl(5,3)=t
bzl(5,3)=((zb-za)/(yb-ya))*(t-byl(5,4))+bzl(5,4)
c
c       Derivatives at fillet-span
c
t=byl2(4)-abs(byl2(4)-byl1(4))/factor2
xa=bxl2d(4)
xb=bxl2(4)
ya=byl2d(4)
yb=byl2(4)
za=bzl2d(4)
zb=bzl2(4)
bxl(2,3)=((xb-xa)/(yb-ya))*(t-byl(2,4))+bxl(2,4)
byl(2,3)=t
bzl(2,3)=((zb-za)/(yb-ya))*(t-byl(2,4))+bzl(2,4)
t=byl2(9)-abs(byl2(9)-byl1(9))/factor2
c       xa=bxl2d(9)
xb=bxl2(9)
xa=bxl2(9)
ya=byl2d(9)
yb=byl2(9)
za=bzl2d(9)
zb=bzl2(9)
bxl(3,3)=((xb-xa)/(yb-ya))*(t-byl(3,4))+bxl(3,4)
byl(3,3)=t
bzl(3,3)=((zb-za)/(yb-ya))*(t-byl(3,4))+bzl(3,4)
t=byl2(16)-abs(byl2(16)-byl1(16))/factor2
xa=bxl2d(16)
xb=bxl2(16)
ya=byl2d(16)
yb=byl2(16)
za=bzl2d(16)
zb=bzl2(16)
bxl(4,3)=((xb-xa)/(yb-ya))*(t-byl(4,4))+bxl(4,4)
byl(4,3)=t
bzl(4,3)=((zb-za)/(yb-ya))*(t-byl(4,4))+bzl(4,4)
c
```

```
c        Derivatives at fillet root
c
t=bzl1(4)+abs(bzl2(4)-bzl1(4))/factor1
xa=bxl1(4)
xb=bxl1d(4)
ya=byl1(4)
yb=byl1d(4)
za=bzl1(4)
zb=bzl1d(4)
bxl(2,2)=(xb-xa)/(zb-za)*(t-bzl(2,1))+bxl(2,1)
byl(2,2)=(yb-ya)/(zb-za)*(t-bzl(2,1))+byl(2,1)
bzl(2,2)=t
t=bzl1(9)+abs(bzl1(9)-bzl2(9))/factor1
xa=bxl1(9)
xb=bxl1d(9)
ya=byl1(9)
yb=byl1d(9)
za=bzl1(9)
zb=bzl1d(9)
bxl(3,2)=(xb-xa)/(zb-za)*(t-bzl(3,1))+bxl(3,1)
byl(3,2)=(yb-ya)/(zb-za)*(t-bzl(3,1))+byl(3,1)
c        byl(3,2)=byl(3,2)/10.00
bzl(3,2)=t
t=bzl1(16)+abs(bzl1(16)-bzl2(16))/factor 1
xa=bxl1(16)
xb=bxl1d(16)
ya=byl1(16)
yb=byl1d(16)
za=bzl1(16)
zb=bzl1d(16)
bxl(4,2)=(xb-xa)/(zb-za)*(t-bzl(4,1))+bxl(4,1)
byl(4,2)=(yb-ya)/(zb-za)*(t-bzl(4,1))+byl(4,1)
bzl(4,2)=t
c
c        do 500 i=1,5
c                do 500 j=1,4
c                        write(6,140) 'The ',i,j,' element is'
c     %                  ,bxu(i,j),byu(i,j),bzu(i,j)
c 500   continue

 120    format(3e20.8)
```

```
 130    format()
 140    format(A,i1,',',i1,A,3e20.8)
call plotsur(bxu,byu,bzu,4,3,20,curves)
open(30,file='surface.dat',form='formatted',status='old')
call system('rm upsurf.dat')
open(31,file='upsurf.dat',form='formatted',status='new')
do 150 i=1,curves
do 150 j=1,20
read(30,120)bxu(i,j),byu(i,j),bzu(i,j)
 150    continue
close(30)
do 160 i=1,curves
if (i.eq.1) then
write(31,120)(bxu1(k),byu1(k),bzu1(k),k=1,20)
else if (i.eq.curves) then
write(31,120)(bxu2(k),byu2(k),bzu2(k),k=1,20)
else
do 170 j=1,20
write(31,120)bxu(i,j),byu(i,j),bzu(i,j)
 170            continue
endif
 160    continue
close(31)
call system('mv patch.dat uppatch.dat')
call plotsur(bxl,byl,bzl,4,3,20,curves)
open(32,file='surface.dat',form='formatted',status='old')
call system('rm losurf.dat')
open(33,file='losurf.dat',form='formatted',status='new')
do 180 i=1,curves
do 180 j=1,20
read(32,120)bxl(i,j),byl(i,j),bzl(i,j)
 180    continue
close(32)
do 190 i=1,curves
if (i.eq.1) then
write(33,120)(bxl1(k),byl1(k),bzl1(k),k=1,20)
else if (i.eq.curves) then
write(33,120)(bxl2(k),byl2(k),bzl2(k),k=1,20)
else
do 200 j=1,20
write(33,120)bxl(i,j),byl(i,j),bzl(i,j)
 200            continue
```

```
endif
 190    continue
close(33)
call system('mv patch.dat lopatch.dat')
end
```

The actual calculation of the coordinates of a suface point is done in routine plotsur which is an examle of the tensor product principle (See Appendix C).

```
subroutine plotsur(bx,by,bz,deg_u,deg_v,u_pts,v_pts)
c
c       This subroutine plots v_pts isoparametric
c  curves of the surface,
c        each with u_pts on it
c
c       input:  bx, by:          arrays with x- and y-coordinates of
c                                control net
c               deg_u,deg_v:     degrees in the u- and v- direction
c               u_pts:           how many points per curve
c               v_pts:           how many curves to draw
c
c       Output:                  Plot of isolines
c The code is a Modified Version of Farin[1990]
integer deg_u, deg_v, u_pts, v_pts
double precision bx(0:20,0:20),by(0:20,0:20), bz(0:20,0:20)
integer i,j,u_pts1
double precision u,v, del_u,del_v
double precision aux_x(0:5),aux_y(0:5),aux_z(0:5)
double precision arrax(0:5),array(0:5),arraz(0:5)
double precision pts_x(0:100), pts_y(0:100),pts_z(0:100)
integer iv

call system('rm surface.dat')
call system('rm patch.dat')
open(3,file='surface.dat',form='formatted',status='new')
del_v=1.0/((v_pts)*1.00)
v=0.0
c
c       first iso-curve is (bx(u,0), by(u,0))
c
```

```
do 10 iv=0,v_pts
do 20 i=0,deg_u
do 30 j=0,deg_v
arrax(j)=bx(i,j)
array(j)=by(i,j)
arraz(j)=bz(i,j)
 30                    continue
call hornbez(deg_v,arrax,v,aux_x(i))
call hornbez(deg_v,array,v,aux_y(i))
call hornbez(deg_v,arraz,v,aux_z(i))
 20             continue
c
c       The bezier points for the iso-curve
c are put in aux_x, and aux_y
c
call bezpt(deg_u,u_pts-1,aux_x,pts_x)
call bezpt(deg_u,u_pts-1,aux_y,pts_y)
call bezpt(deg_u,u_pts-1,aux_z,pts_z)
c
c       Do something like saving onto a file
c or initiating a gnuplot session
c something in the order of the following:
c
c       move_abs_2(pts_x(0),pts_y(0))
c       polyline_abs_2(pts_x,pts_y,u_pts1)
c
do 50 ii=0,19
write(3,100) pts_x(ii),pts_y(ii),pts_z(ii)
 50     continue
 100    format(3e20.8)
c       write(3,110)
 110    format()
v=v+del_v
 10     continue
close(3)
open(4,file='patch.dat',form='formatted',status='new')
do 105 j=0,3
do 106 i=0,4
write(4,120)bx(i,j),by(i,j),bz(i,j)
c       write(6,120)bx(i,j),by(i,j),bz(i,j)
 106    continue
c       write(4,110)
```

```
 105    continue
 120    format(3e20.8)
close(4)
close(3)
return
end
```

After the completion of the `testplot` routine, the volume constraints by the landing gear are read in. Routine `lgcheck` calculates the landing gear requirements for the given aircraft configuration, reads in the fillet surface definition, the wing surface definition from the original run of the surface generation code, and compares the volume required with the volume available. If the volume available is not sufficient to accomodate the landing gear as calculated in Chapter 5 a message is printed and the appropriate parameters for an updated lower wing surfacd and fillet calculation are stored in the file `lgconstraints.coeffs`. This file then is read by the `JOINBEZ` routine and the fillet surface is re-evaluated.

When all landing gear constraints are satisfied, the fillet surface points are read in by the programs `up2eagle` and `lo2eagle` and the appropriate surface generation code is produced and appended to the aircraft surface generation code. The two programs, `up2eagle` and `lo2eagle` are very similar, and therefore only one listing will be presented:

```
 program UP2EAGLE

      double precision ptx1,ptx2,pty1,pty2,ptz1,ptz2
integer core,i,j,curves,core1
curves=5
core=450
core1=470

open (3,file='upsurf.dat',form='formatted', status='old')
call system('rm up2eagle.srf')
open (4,file='up2eagle.srf',form='formatted',status='new')
```

```
do 10 i=1,curves+1
write(6,*),'curve=',i
do 20 j=1,20
if (j.eq.1) then
read(3,100),ptx1,pty1,ptz1
elseif (j.gt.1) then
read(3,100),ptx2,pty2,ptz2
write(4,110),ptx1,pty1,ptz1,ptx2,pty2,ptz2,core
ptx1=ptx2
pty1=pty2
ptz1=ptz2
core=core+1
endif
 20              continue

core=450
c                read(3,120)
write(4,130)core1
write(4,140)
core1=core1+1
 10     continue
write(4,150)core1
write(4,160)
write(4,170)
write(4,171)
write(4,172)
write(4,173)
write(4,174)
write(4,175)
write(4,180)
write(4,182)
write(4,184)
 100    format(3e20.8)
 110    format(27h$'current',points=2,values=,6(f10.2,','),
     %                    8hcoreout=,i3,1h$)
 120    format()
 130    format(17h$'concat',corein=,8h450,-468,
     %                  22h,edge='first',coreout=,i3,1h$)
 140    format(1h*)
 150    format(19h$'assemble',corein=,8h470,-475,10h,points=20,
     %                   22h,change='yes',coreout=,i3,1h$)
 160    format(47h$'surdist',corein=476,points=20,10,coreout=476$)
```

```
170     format(43h$'blend',bound=475,92,curves=4,coreout=475$)
171     format(42h$'blend',bound=92,133,curves=3,coreout=92$)
172     format(44h$'blend',bound=133,174,curves=3,coreout=133$)
173     format(44h$'blend',bound=174,176,curves=3,coreout=174$)
174     format(44h$'blend',bound=176,178,curves=3,coreout=176$)
175     format(44h$'blend',bound=178,180,curves=2,coreout=178$)
180     format(44h$'blend',bound=180,182,curves=2,coreout=180$)
 182    format(44h$'blend',bound=182,184,curves=3,coreout=182$)
 184    format(44h$'concat',corein=476,475,92,133,174,176,178,,
     %              34h180,182,edge='second',coreout=300$)
close(4)
close(3)
end
```

F.1.3 Aircraft Geometry Modification

The surface code segments are then inserted into the surface generation code with programs similar to the GAWK program `isectu1.awk`. The EAGLE surface generation code is then run once more to produce the geometry definition of the aircraft as modified after the fillet insertion. Upon successful completion of the execution, the grid generation program is executed (See Appendix B) which then produces an output file with the grid point coordinates. This file is read by the Thin-Layer Navier-Stokes (TLNS) solver along with an input definition file (See Appendix D). The completion of the TLNS run completes the Analysis Stage of the thesis.

F.2 Optimization Stage

When the analysis is completed, the data obtained from the Table of Experiments is used to produce a second order response surface. In this case, the response surface was constructed using MACANOVA, a program for statistical analysis and matrix manipulation. The results in ratios of the original aircraft

configuration with no fillet for the six variables mentioned in Chapter 6 are shown in Table F.1.

Using MACANOVA, the mean was subtracted from every entry, and the surface fitting was then performed. MACANOVA contains a macro called `rscanon` which fits a second order surface to the available data. The results are as follow:

The constant term: $c = 44.905$

The linear terms:

$$\begin{aligned} x_1 &= -0.093292 \\ x_2 &= -0.096064 \\ x_3 &= -0.040595 \\ x_4 &= -28.416 \\ x_5 &= -16.247 \\ x_6 &= -1.573 \end{aligned}$$

The quadratic and interaction terms:

$$\begin{bmatrix} 0.0020289 & -0.00091602 & 0.0 & -0.021468 & -0.0096275 & 0.0062781 \\ -0.000916 & 0.0020402 & 0.0 & -0.021757 & 0.0097442 & 0.0066748 \\ 0.0 & 0.0 & 0.000161 & 0.0017259 & -0.0018593 & 0.00012172 \\ -0.021468 & -0.021757 & 0.0017259 & 6.9372 & 2.8286 & -0.33005 \\ 0.0096275 & 0.0097442 & -0.001859 & 2.8286 & 1.2806 & 0.22908 \\ 0.0062781 & 0.0066748 & 0.0001217 & -0.33005 & 0.22908 & 0.10499 \end{bmatrix}$$

The stationary point: $y_0 = -1.4464$ for the following values

$$x_1 = -0.76376$$

Design 628A Results	
Trial	Objective
1	0.9387
2	2.0374
3	2.0593
4	2.2407
5	2.2556
6	0.7267
7	0.7410
8	0.73519
9	0.72778
10	1.6327
11	1.6537
12	0.5685
13	1.6481
14	2.5056
15	2.5252
16	2.4893
17	2.5037
18	1.8333
19	1.8352
20	1.8389
21	1.9426
22	2.2389
23	2.2463
24	2.2852
25	1.8556
26	1.8278
27	1.8148
28	0.5482

Table F.1: Results Table

$$x_2 = -1.0931$$

$$x_3 = 134.62$$

$$x_4 = 1.3677$$

$$x_5 = 2.3461$$

$$x_6 = 6.6313$$

These values do not all lie within the feasible region. Such values for the first variables would produce fillet geometries that would lie within the original wing, or would lie within the fuselage. Since the stationary point lies outside the feasible region, the NPSOL optimizer is used once more. The code that performs the actual optimization is listed below:

```
      IMPLICIT           DOUBLE PRECISION(A-H,O-Z)

*     Set the declared array dimensions.
*     NROWA  = the declared row dimension of  A.
*     NROWJ  = the declared row dimension of  CJAC.
*     NROWR  = the declared row dimension of  R.
*     MAXN   = maximum no. of variables allowed for.
*     MAXBND = maximum no. of var+ lin & nonlin constr
*     LIWORK = length of the integer work array.
*     LWORK  = length of the double precision work array.

      PARAMETER (NROWA  =  0, NROWJ  =  0, NROWR =   20,
     $           MAXN   =  6, LIWORK =  70, LWORK = 1000,
     $           MAXBND =  MAXN + NROWA + NROWJ)

      INTEGER            ISTATE(MAXBND)
      INTEGER            IWORK(LIWORK)
      DOUBLE PRECISION   A(NROWA,MAXN)
      DOUBLE PRECISION   BL(MAXBND), BU(MAXBND)
      DOUBLE PRECISION   C(NROWJ), CJAC(NROWJ,MAXN)
      DOUBLE PRECISION   CLAMDA(MAXBND)
      DOUBLE PRECISION   OBJGRD(MAXN), R(NROWR,MAXN)
      DOUBLE PRECISION   X(MAXN)
      DOUBLE PRECISION   WORK(LWORK)
```

```
      EXTERNAL           OBJFN1

      PARAMETER          (ZERO = 0.0, ONE = 1.0)

*     Set the actual problem dimensions.
*     N      = the number of variables.
*     NCLIN=number of general lin constr (may be 0).
*     NCNLN=number of nonlin constr (may be 0).

      N      = 6
      NCLIN  = 0
      NCNLN  = 0
      NBND   = N + NCLIN + NCNLN

*     ----------------------------------
*     Assign file numbers and the data arrays.
*     NOUT   = the unit number for printing.
*     IOPTNS = the unit number for reading the options file.
*     Bounds.ge.BIGBND  will be treated as plus  infinity.
*     Bounds.le.- BIGBND  will be treated as minus infinity.
*     A      = the linear constraint matrix.
*     BL     = the lower bounds on  x,  a'x  and  c(x).
*     BU     = the upper bounds on  x,  a'x  and  c(x).
*     X      = the initial estimate of the solution.
*     ----------------------------------
      NOUT   = 6
      IOPTNS = 5
      BIGBND = 1.0D+15

*     Set the matrix  A.

      DO 40 J = 1, N
 DO 30 I = 1, NCLIN
    A(I,J) = ZERO
   30    CONTINUE
   40 CONTINUE
      A(1,1) = -ONE
      A(1,2) =  ONE
      A(2,2) = -ONE
      A(2,3) =  ONE
      A(3,3) =  ONE
      A(3,4) = -ONE
```

```
      A(4,4) =  ONE
      A(4,5) = -ONE

*     Set the bounds.

      DO 50 J  =  1, NBND
 BL(J) = -BIGBND
 BU(J) =  BIGBND
   50 CONTINUE
      BL(1)  =  2.0
      BL(2)  =  2.0
      BL(3)  =  2.0
      BL(4)  =  1.1
      BL(5)  =  1.1
      BL(6)  =  3.3

      BU(1)  =  50.0
      BU(2)  =  50.0
      BU(3)  =  200.00
      BU(4) =   2.0
      BU(5) =   3.0
      BU(6) =   6.5
*     Set lower bounds of zero for all four lin constr

*     Set upper bounds of one for all 14 nonlin= constr


*     Set the initial estimate of  X.

      X(1)   =  38.0
      X(2)   =  38.0
      X(3)   =  153.0
      X(4)   =  1.55
      X(5)   =  1.725
      X(6)   =  5.2

*     ------------------------------
*     Read the options file.
*     ------------------------------

      CALL NPFILE( IOPTNS, INFORM )
      IF (INFORM .NE. 0) THEN
```

```
      WRITE (NOUT, 3000) INFORM
      STOP
           END IF

*     -----------------------------------
*     Solve the problem.
*     -----------------------------------

      CALL NPSOL ( N, NCLIN, NCNLN, NROWA, NROWJ, NROWR,
     $             A, BL, BU,
     $             CONFN1, OBJFN1,
     $             INFORM, ITER, ISTATE,
     $             C, CJAC, CLAMDA, OBJF, OBJGRD, R, X,
     $             IWORK, LIWORK, WORK, LWORK )

      IF (INFORM .GT. 0) GO TO 900



      DO 100 J = 1, N
X(J)  = X(J) + 0.1
  100 CONTINUE

*     The previous parameters are retained and updated.

      CALL NPOPTN( '   Derivative level                    0')
      CALL NPOPTN( '   Verify                             No')
      CALL NPOPTN( '   Warm Start')
      CALL NPOPTN( '   Major iterations                   20')
      CALL NPOPTN( '   Major print level                  10')

      CALL NPSOL ( N, NCLIN, NCNLN, NROWA, NROWJ, NROWR,
     $             A, BL, BU,
     $             CONFN2, OBJFN2,
     $             INFORM, ITER, ISTATE,
     $             C, CJAC, CLAMDA, OBJF, OBJGRD, R, X,
     $             IWORK, LIWORK, WORK, LWORK )

      IF (INFORM .GT. 0) GO TO 900
      STOP

*     -----------
```

```
*     Error exit.
*     -----------

  900 WRITE (NOUT, 3010) INFORM
      STOP

 3000 FORMAT(/ ' NPFILE terminated with  INFORM =', I3)
 3010 FORMAT(/ ' NPSOL  terminated with  INFORM =', I3)

*     End of the example program for NPSOL.

      END
*++++++++++++++++++++++++++++++++++++
      SUBROUTINE OBJFN1( MODE, N, X, OBJF, OBJGRD, NSTATE )
      IMPLICIT           DOUBLE PRECISION(A-H,O-Z)
      DOUBLE PRECISION   X(N), OBJGRD(N)

*-----------------------------------
*     OBJFN1  computes the value and first
*     derivatives of the nonlinear
*     objective function.
*-----------------------------------

      c=44.905
      x1=-0.093292
      x2=-0.096064
      x3=-0.040595
      x4=-28.416
      x5=-16.247
      x6=-1.573
      x11=0.0020289
      x12=-0.00091602
      x13=0.
      x14=-0.021468
      x15=0.0096275
      x16=0.0062781
      x21=-0.00091602
      x22=0.0020402
      x23=0.0
      x24=-0.021757
      x25=0.0097442
      x26=0.0066748
```

```
      x31=0.0
      x32=0.0
      x33=0.0001609
      x34=0.0017259
      x35=-0.0018593
      x36=0.00012172
      x41=-0.021468
      x42=-0.021757
      x43=0.0017259
      x44=6.9372
      x45=2.8286
      x46=-.33005
      x51=0.0096275
      x52=0.0097442
      x53=-0.0018593
      x54=2.8286
      x55=1.2806
      x56=0.22908
      x61=0.0062781
      x62=0.0066748
      x63=0.00012172
      x64=-0.33005
      x65=0.22908
      x66=0.10499
      OBJF=c+X(1)*x1+X(2)*x2+X(3)*x3+X(4)*x4+X(5)*x5+X(6)*x6+
     $     X(1)**2*x11+X(2)**2*x22+X(3)**2*x33+X(4)**2*x44+
     $     X(5)**2*x55+X(6)**2*x66+
     $X(1)*X(2)*x12+X(1)*X(3)*x13+X(1)*X(4)*x14+X(1)*X(5)*x15
     $      +X(1)*X(6)*x16+
     $X(2)*X(1)*x21+X(2)*X(3)*x23+X(2)*X(4)*x24+X(2)*X(5)*x25
     $      +X(2)*X(6)*x26+
     $X(3)*X(1)*x31+X(3)*X(2)*x32+X(3)*X(4)*x34+X(3)*X(5)*x35
     $      +X(3)*X(6)*x36+
     $X(4)*X(1)*x41+X(4)*X(2)*x42+X(4)*X(3)*x43+X(4)*X(5)*x45
     $      +X(4)*X(6)*x46+
     $X(5)*X(1)*x51+X(5)*X(2)*x52+X(5)*X(3)*x53+X(5)*X(4)*x54
     $      +X(5)*X(6)*x56+
     $X(6)*X(1)*x61+X(6)*X(2)*x62+X(6)*X(3)*x63+X(6)*X(4)*x64
     $      +X(6)*X(5)*x65

      RETURN
```

```
*       End of  OBJFN1.

        END
```

The result of the optimization is summarized in the following list:

$$
\begin{aligned}
x_1 &= 33.4 \\
x_2 &= 33.6 \\
x_3 &= 112.5 \\
x_4 &= 2.0 \\
x_5 &= 1.1 \\
x_6 &= 6.5
\end{aligned}
$$

The value computed by the Response Surface produces a ratio of 0.51, which is the lowest when compared with the figures in Table F.1. When the optimum design variable values are entered to the optimization loop, after the execution of the EAGLE TLNS code, the result is a ratio of .49.

Appendix G

Figures and Graphs

G.1 Description

This appendix contains sample files indicative of the variety of the results that were obtained through the course of the simulation runs.

The first five figures describe the pressure distribution on five stations along the span of the wing, for a fillet configuration producing high drag [1].

It must be noted that the same color shades in the rendered figures do not, as a rule, represent similar values of the pressure coefficient. In fact, in a single rendered image they will represent a number of ranges of pressure coefficients. The colormap for the rendering was created this way to pronounce the different ranges of pressure coefficients and make more apparent the pressure gradients on the surface of the wing. Nevertheless, the colormap used for all the cases presented here is the same. Therefore, if two wing surfaces exhibited exactly the same pressure distribution, they would trigger the rendering of exactly the same pattern.

---

[1]The estimation of the pressure coefficient had come across a few ill-behaved plotting routines in the past, which rather than displaying the correct value of the pressure, kept a "running total" and therefore resulting in unnaturally high pressure coefficients. This problem has now been resolved, and all the routines are believed to function correctly

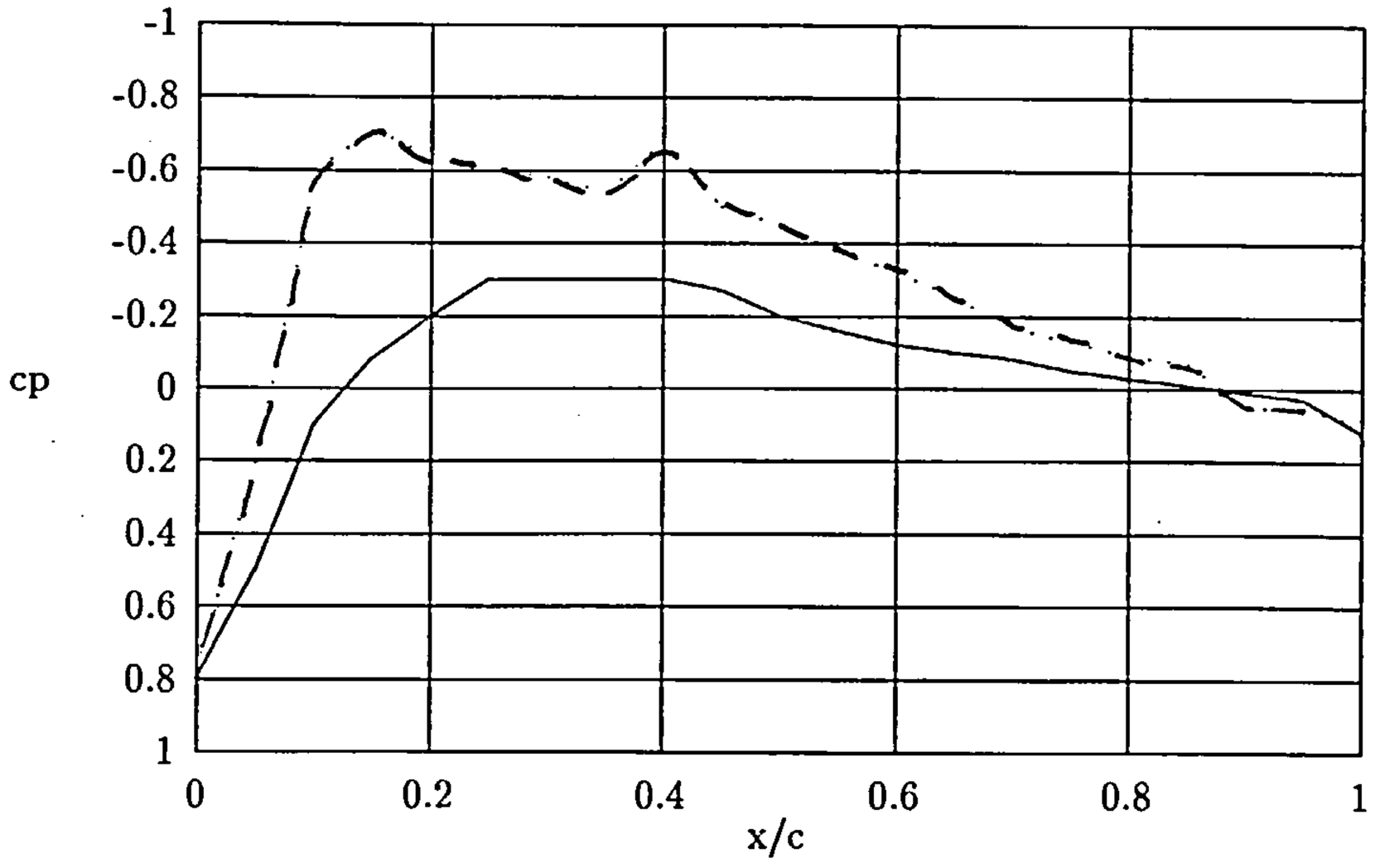

Figure G.1: c_p distribution at $Y/S = 0$ for high drag fillet

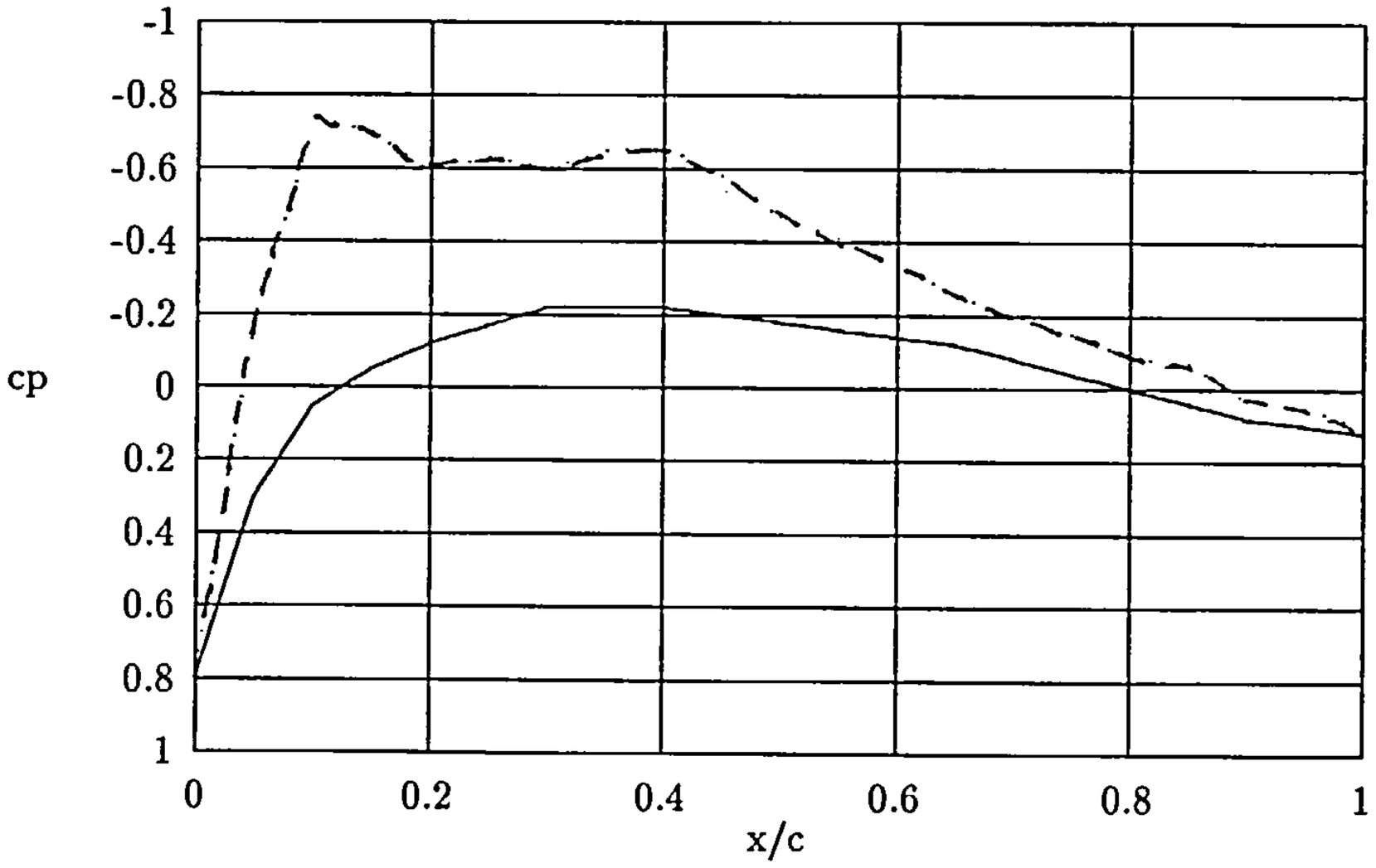

Figure G.2: c_p distribution at $Y/S = 0.1$ for high drag fillet

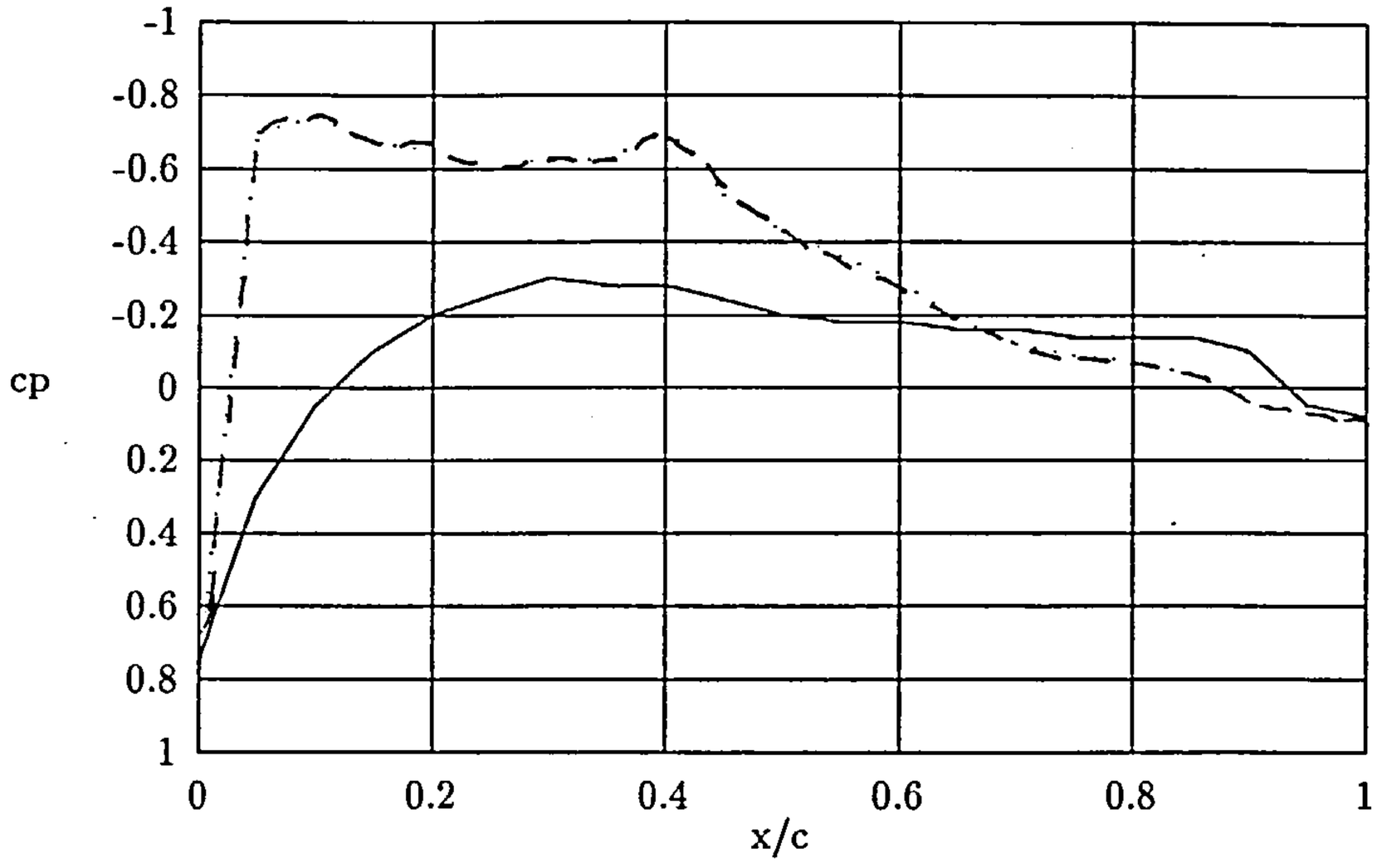

Figure G.3: c_p distribution at $Y/S = 0.5$ for high drag fillet

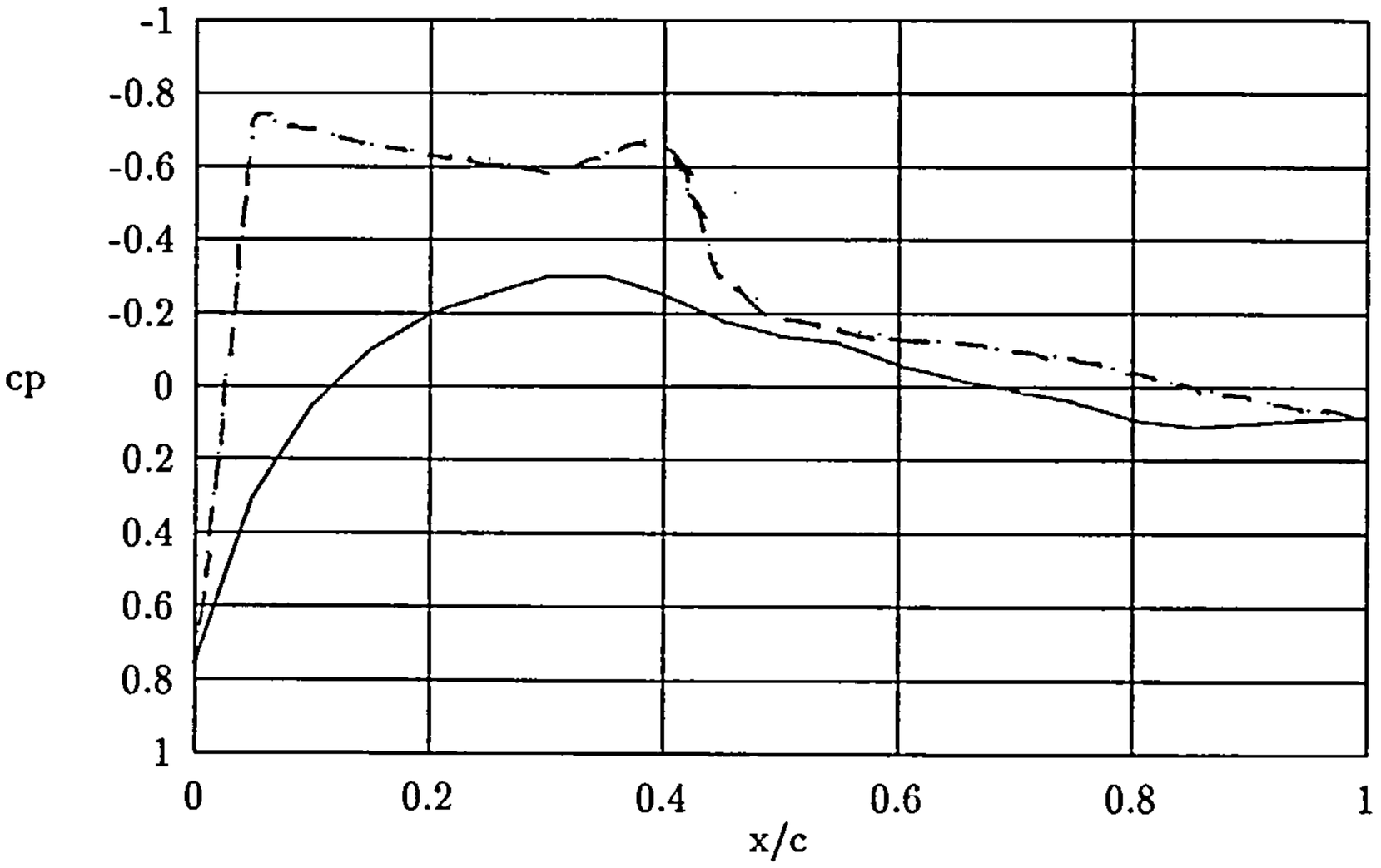

Figure G.4: c_p distribution at $Y/S = 1$ for high drag fillet

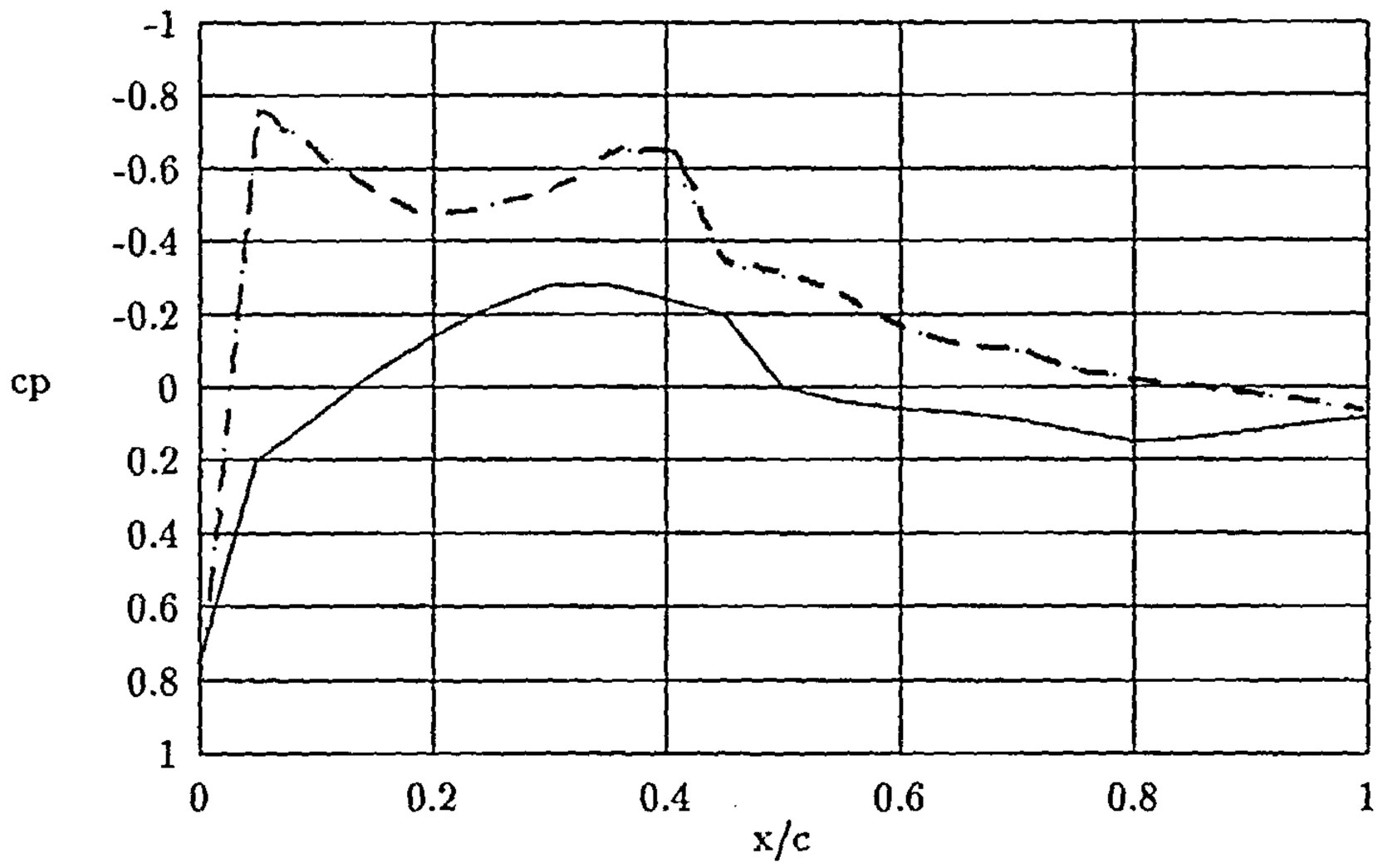

Figure G.5: c_p distribution at $Y/S = 3$ for high drag fillet

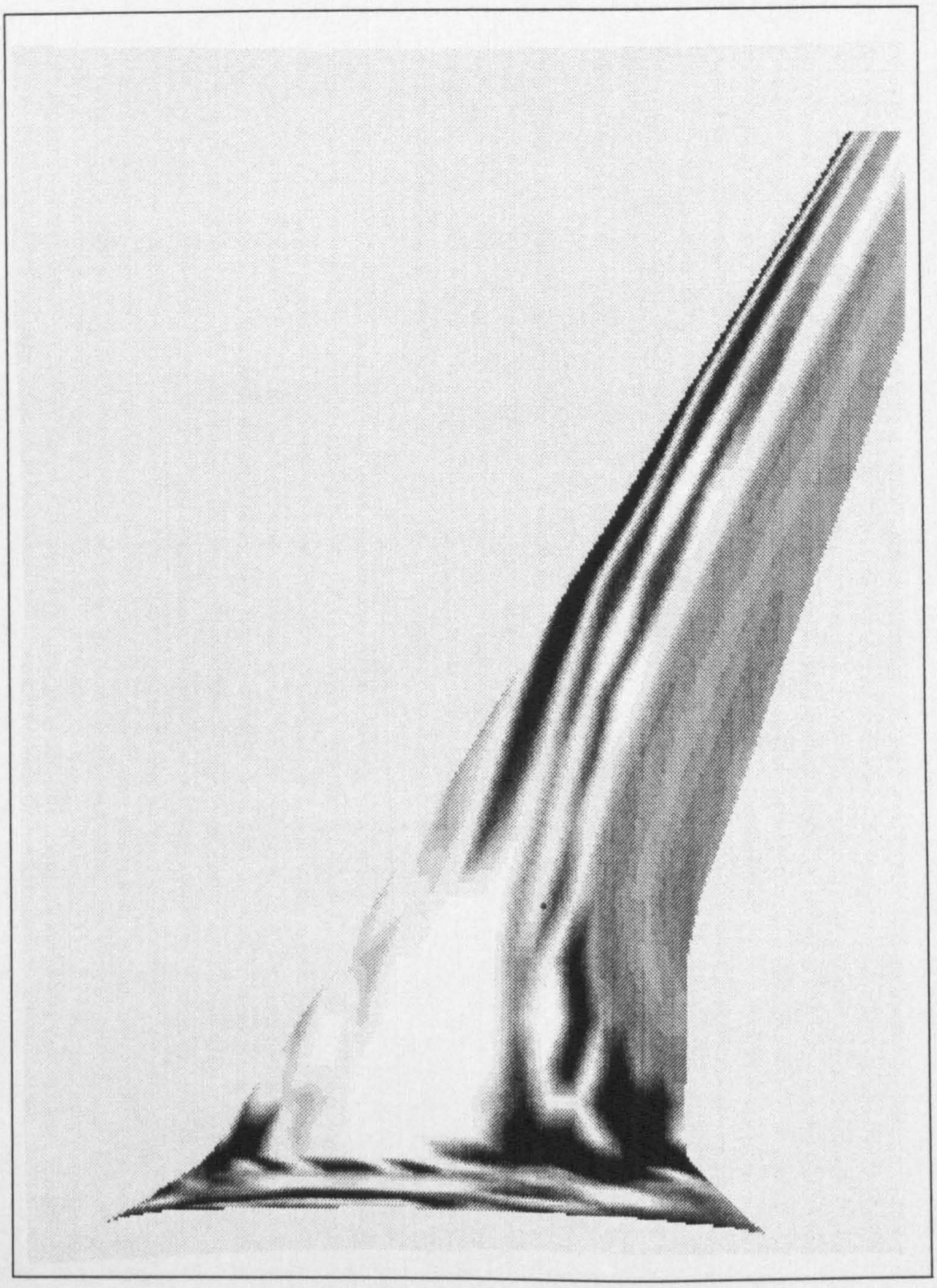

Figure G.6: Rendered Upper Surface Pressure Distribution for High Drag Configuration